UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

SISTEMA DE REGA INTELIGENTE

CARLOS JORGE VELEZ MÃO DE FERRO

PROJETO

MESTRADO EM INFORMÁTICA

2013

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

SISTEMA DE REGA INTELIGENTE

CARLOS JORGE VELEZ MÃO DE FERRO

PROJETO

MESTRADO EM INFORMÁTICA

Projeto orientado pelo Prof. Doutor Francisco Cipriano da Cunha Martins

2013

Agradecimentos

Ao meu orientador Francisco Martins, responsável indiscutível pelo meu sucesso e a quem devo a oportunidade de ter realizado este projeto. Foi e continuará a ser para mim um exemplo incontestável da vontade de aprender e partilhar o conhecimento. Com ele aprendi o verdadeiro significado de ser cientista, ou seja, questionar o mundo que nos rodeia e devorar o conhecimento como um músico domina o seu instrumento. Acima de tudo, ele é a prova da relação de camaradagem que pode existir entre professor e aluno. Deixo o voto sincero para que possamos trabalhar muitas mais vezes no futuro.

Ao Filipe Bragança, que será conhecido nesta tese como "cliente", deixo o meu forte agradecimento pois sem ele este projeto não existiria. O Filipe tomou um passo pouco comum em Portugal ao confiar a sua ideia a um grupo de investigadores da faculdade sendo por isso um pioneiro. O sucesso deste projeto é a prova viva de que é possível aproximar o meio universitário à Indústria e à resolução de problemas concretos do quotidiano.

Às professoras Ana Luísa Respício, Beatriz Carmo e Graça Gaspar por me ensinarem a ser aluno e professor ao mesmo tempo e confiarem sempre no meu potencial.

Ao professor Luís Carriço pelo seu apoio durante a execução deste projeto.

À professora Guiomar Evans, a quem agradeço as tão preciosas sessões de Eletrónica sem as quais seria extremamente difícil aventurar-me nesse mundo e pelas suas revisões técnicas a este trabalho.

Um agradecimento muito forte à professora Maria Luísa Paramés pela sua motivação, por provar que é possível conciliar disciplina e responsabilidade sem deixar de ser divertido aprender e finalmente por me ensinar a nunca desperdiçar uma oportunidade e a acreditar em mim. Sem ela não poderia dizer que fiz a cadeira de Física Experimental.

À minha família, em especial aos meus pais, deixo o agradecimento mais profundo. Na verdade, uma boa forma de contabilizar esse agradecimento seria contar o número de transístores do meu Macbook Air. Felizmente estamos em 2013!! Sinto-me realmente um privilegiado por tanto amor e dedicação que sempre recebi. São sem dúvida o meu fio condutor que terei para sempre um gosto enorme em seguir. À minha irmã deixo um agradecimento especial por me mostrar que os irmãos mais novos também têm muito para ensinar aos mais velhos.

À Ana Saramago, por ser especial, agradeço a sua paciência e carinho que teve para me aturar nos momentos melhores e piores desta tese. O que seria de mim senão a tivesse

conhecido naquela famosa saída de campo! De uma coisa estou certo. Por mais universos paralelos que possam existir todos eles acabariam por, mais tarde ou mais cedo, dar-me a conhecer a Ana e o amor que temos um pelo outro. Deixo ainda um agradecimento à sua família que me emprestou a casa onde fiz provavelmente metade do meu trabalho.

Sinto-me grato por tantos amigos que fiz ao longo da minha vida académica. Contudo, não posso deixar de fazer um agradecimento especial ao Diogo Catita, Tiago Silvestre, Miguel Aroucha, João Francisco André, Ruben Campos, Rui Pires, Bruno Valente, Miguel Garcia, Gonçalo Gomes e um agradecimento especialíssimo ao Nuno Dias Martins pelos anos como colega de trabalho, como visionário, como amigo sincero e por nunca deixar de acreditar no Benfica! A todos eles desejo o maior sucesso.

Ao centro Teclabs por financiar a minha bolsa e ao projeto MACAW PTDC/EIA-EIA/115730/2009 da FCT pelo suporte financeiro parcial da investigação.

Ao Maurício Martins (Leds & Chips) agradeço os seus ensinamentos sobre o mundo do Arduino e disponibilidade como fornecedor principal de *hardware*. Agradeço ainda ao Edward Huang (Quectel) pela disponibilidade na resolução de questões relacionadas com o módulo GSM/GPRS.

Finalmente, deixo um agradecimento especial ao corpo do Departamento de Informática e à Faculdade de Ciências em geral que me ofereceram todas as condições para realizar o meu trabalho e que me fazem acreditar no ensino de qualidade em Portugal.

Dedico este trabalho à minha mãe por ser a primeira da família a escrever uma tese. Deixo ainda uma dedicatória sentida à minha gata que faleceu no dia 3 de Setembro de 2013 e que tantas vezes foi a minha fiel companheira de estudo.

Resumo

A utilização de sistemas de rega pelo ser humano remonta às primeiras populações que adotaram um estilo de vida sedentário em vez de um estilo nómada. Desde então, as técnicas utilizadas diversificaram-se indo ao encontro das necessidades de cada época.

Independentemente das técnicas utilizadas e do propósito da rega (e.g., lazer ou agricultura), a racionalização dos recursos hídricos foi desde cedo uma prioridade. Essa racionalização exigia normalmente a supervisão humana para controlo e ajustamento da quantidade de água utilizada. Além disso, esse controlo era baseado em experiência e intuição e poderia nem sempre revelar-se o mais adequado ao tipo de plantas.

Nas últimas décadas tem-se assistido a uma tendência para automatizar os sistemas de rega, reduzindo a mão de obra necessária para a sua manutenção, e mais recentemente à utilização de sensores para monitorizar as necessidades de cada planta. Apesar de existirem já diversos sistemas automáticos em vigor, muitos deles obrigam ainda assim à programação local dos controladores de rega. Se considerarmos a gestão de centenas de jardins, essa tarefa pode revelar-se dispendiosa, muito em parte devido ao tempo despendido e às despesas de deslocação. Por outro lado, muitos dos controladores que permitem a comunicação remota não se preocupam com o consumo energético pois têm acesso à rede elétrica. Em muitos casos onde não é possível ter acesso à rede, torna-se fundamental repensar o rendimento energético do controlador já que a comunicação remota exige mais energia.

O Sistema de Rega Inteligente (SRI) é um produto constituído por vários controladores e por uma plataforma de gestão. Os controladores têm como características principais a elevada autonomia energética, facilidade de instalação, compatibilidade com as infraestruturas de rega já existentes (válvulas solenoides), execução de um plano de rega automático, conetividade com vários sensores e gestão e reprogramação remotas sem fios. A plataforma de gestão permite controlar e monitorizar a rega de vários jardins em simultâneo recolhendo periodicamente diversas informações de cada controlador e que serão úteis para melhorar o plano de rega de cada jardim. O SRI é orientado ao controlo de rega em jardins municipais mas vários conceitos poderão ser facilmente expandidos para outras utilizações, em particular, na agricultura.

Para satisfazer os requisitos referidos, desenvolvemos uma solução de *software* constituída por uma plataforma de gestão que utiliza a tecnologia *Java Enterprise Edition*

[Ora13b] e que está instalada no sistema de *nuvem* da *Amazon* [Ama]. Adicionalmente, concebemos um circuito elétrico de raiz, que nos permitirá produzir uma placa de circuito impressa feita à medida. O resultado é um controlador que permite comunicação remota sem fios e ao mesmo tempo altamente eficiente.

Palavras-chave: sistemas de rega, microcontroladores, circuitos elétricos, controlo remoto sem fios, aplicações empresariais

Abstract

Irrigation systems have been used by human beings since sedentary lifestyle took over their nomad habits. Since then, a multitude of techniques were developed to meet the requirements of each epoch.

Regardless of the technique used or of the irrigation purpose (e.g., leisure or agriculture), the water savings were, undoubtedly, one of the main concerns. This concern required, typically, the human supervision for control and adjustment of the quantity of water used. Moreover, that control was mainly based on experience and intuition and many times revealed itself inadequate for plants' needs.

On the last decades, efforts were made to automate the irrigation systems, reducing human labor. Also, recently, different kinds of sensors were introduced to monitor each different plant kind. Although several automatic systems are in use today, many still require a local programming for each controller. If one considers the management of hundreds of gardens, the water scheduling task can be extremely costly for the time and itineration costs it demands. Moreover, many controllers which support remote communication don't care about energy consumption since they have access to public electric network. Where it is not feasible to have this energy source, one must rethink the way controllers are designed regarding their consumption. This is particularly important because remote communication has strong energy requirements.

SRI (Smart Irrigation System) is a product comprising several controllers and a management platform. Controllers are characterized by a high energy efficiency, ease of installation, compatibility with existent water infrastructures (solenoid valves), execution of an automatic irrigation plan, support for several sensors and support for remote wireless management and reprogramming. The management platform offers the possibility to control and monitorize the irrigation in several gardens simultaneously and gather periodically data, useful for improving the irrigation plan of each specific garden. SRI is public gardens oriented, but several concepts can be easily reused for other purposes, like agriculture.

In order to satisfy the client requirements, we developed a software solution composed by a management platform which is build on top of the Java Enterprise Edition [Ora13b] technology. This platform is installed on the Amazon *cloud* system [Ama]. Furthermore, we designed an electric circuit from scratch to be used on a custom made printed circuit

board. The result is a controller capable of remote wireless communication but also highly efficient.

Keywords: irrigation systems, microcontrollers, electronic circuits, remote wireless control, enterprise applications

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

> *"You can't just ask customers what they want and then try to give that to them. By the time you get it built, they'll want something new."*
>
> (Steve Jobs, 1 de Abril de 1989)

O trabalho que se apresenta nasceu de uma necessidade real, que nos foi proposta por um profissional de rega e construção de jardins públicos. Neste capítulo introdutório enunciamos as razões que nos motivaram a desenvolver este projeto e quais as suas contribuições. Em seguida resumimos o trabalho efetuado e terminamos com uma breve descrição da estrutura desta tese.

1.1 Motivação

Atualmente a instalação de jardins urbanos encontra-se disseminada um pouco por todo o mundo. O desejo de aproximação da comunidade aos espaços verdes, último reduto de lazer e refúgio de ar puro das grandes cidades, incentiva fortemente a criação de jardins em locais públicos. Estes espaços verdes têm tamanhos e características diferentes e, regra geral, possuem um sistema de rega automático. Contudo, na maioria dos casos, é necessária a deslocação de técnicos ao local para ajustar o planeamento da rega, o que pode significar a deslocação a centenas de jardins. Adicionalmente, a escassez de funcionalidades de muitos dos sistemas já instalados contribui fortemente para o desperdício de água que se deve, em grande parte, à negligência das condições atmosféricas e características específicas de cada jardim, sendo a rega praticamente idêntica em todos eles. Este desperdício tem consequências graves em termos financeiros e agrava a pegada ecológica relativa ao consumo de água potável [Dum12, Eco13].

Com o projeto SRI pretendemos oferecer um sistema para controlar de forma remota a rega em jardins. Em qualquer altura, a partir de qualquer lugar, será possível gerir e monitorizar um plano de rega otimizando-o às necessidades de cada jardim. A monitorização de diversos parâmetros, como humidade do solo, pressão da água e duração da rega permite uma rega mais eficiente, poupando a utilização de recursos hídricos e, consequen-

temente, contribuindo para a redução de gastos com a rega. Estes dados têm um papel importante para melhorar os planos de rega que tenderão a ser mais precisos à medida que se colecione informação histórica.

Nesta tese apresentamos uma plataforma *Web* e um equipamento, o controlador de rega. O desenvolvimento deste controlador envolve a parte de programação do microcontrolador e o desenho do circuito eletrónico do produto que integra um módulo de comunicação GPRS e permite a ligação externa de vários sensores.

As principais contribuições deste trabalho são:

- uma pesquisa do estado da arte que inclui patentes registadas e soluções comercializadas relacionadas com o nosso produto;
- um protótipo funcional de *hardware* capaz de receber instruções de uma aplicação remota e que justifica o desenvolvimento de um produto final;
- o desenho e início de desenvolvimento de um produto final de *hardware*, o controlador SRI, com alta eficiência energética;
- uma solução de *software*, a plataforma de gestão central, que permite comunicar com os controladores SRI e executar a rega remota;
- a entrada de um pedido de patente provisório com vista a proteger as ideias que constituem a nossa solução;
- preparação do produto para comercialização.

1.2 Planeamento

O objetivo inicial deste projeto pressupunha a construção de um protótipo de *hardware* capaz de ser testado num ambiente controlado e a implementação de uma aplicação *Web* que comunicasse com o protótipo. Dado o sucesso na construção deste protótipo, quisemos enriquecer o projeto e iniciámos o desenvolvimento de um produto final de *hardware* comercializável. A aplicação *Web* sofreu também melhorias suportando mais funcionalidades do que aquelas inicialmente propostas pelo cliente. O facto de trabalharmos com um cliente que pretende comercializar este produto, alertou-nos para a forte pressão do mercado e motivou um desenvolvimento mais acelerado. Estes fatores obrigaram, inevitavelmente, à revisão do planeamento inicialmente previsto. Em baixo resumimos o planeamento seguido:

- 10 de Setembro de 2012 – 18 de Outubro de 2012: reunião com o cliente e equipa do TECLABS; levantamento do estado da arte, estudo de sistemas de irrigação existentes no mercado, análise de requisitos e estudo de microcontroladores, sensores e consumos energéticos;

- 18 de Outubro de 2012 – 16 de Novembro de 2012: escrita de casos de uso; início da implementação da plataforma central de gestão;

- 16 de Novembro de 2012 – 18 de Janeiro de 2013: escrita do relatório preliminar; testes da máquina virtual Callas; início do desenvolvimento com o controlador Arduino Uno;

- 18 de Janeiro de 2013 – 15 de Março de 2013: aquisição de *hardware*: sensores, painel solar, módulo GPRS; integração do Arduino Uno com sensores e módulo GPRS; início do desenvolvimento da biblioteca GPRS; continuação do desenvolvimento da plataforma central; demonstração ao cliente das funcionalidades já implementadas no controlador; revisão dos casos de uso;

- 15 de Março de 2013 – 15 Abril de 2013: reunião com o cliente e equipa do TECLABS: demonstração do protótipo em funcionamento com utilização da plataforma central; aquisição do Arduino Pro Mini e experiências para medir o consumo energético; aperfeiçoamento da biblioteca GPRS; aperfeiçoamento da plataforma central: módulo de comunicação com o controlador; experiências com o microcontrolador ATMEGA328P num *breadboard*; testes básicos de consumo; reunião com o cliente e equipa do TECLABS: pesquisa do Estado da Técnica e proteção do produto;

- 15 de Abril de 2013 – 24 de Maio de 2013: início do desenho do circuito eletrónico final; continuação do desenvolvimento do *software* do controlador: máquina de estados, interrupções, modo *sleep*; aperfeiçoamento da plataforma central: segurança, gestão dos planos de rega; aquisição de *hardware*: microcontrolador ATMEGA644, módulo GPRS M10 e experiências com o material adquirido; instalação da plataforma central num servidor Amazon EC2; reunião com o cliente e equipa do TECLABS: balanço do trabalho desenvolvido e início do registo provisório de patente;

- 24 de Maio de 2013 – 24 de Julho de 2013: conclusão do desenho do circuito: alimentação, microcontrolador ATMEGA644 e módulo GPRS M10; conclusão do desenvolvimento da plataforma central e do controlador; início da escrita da Tese;

1.3 Estrutura do documento

Este documento começa por apresentar a visão do projeto, onde se expõem os objetivos iniciais, uma pesquisa de mercado de produtos relacionados, a identificação dos problemas na gestão de rega e de como a nossa solução pode ser vantajosa. A estrutura deste documento reflete o desenvolvimento das duas peças, uma de *software* e outra de *hardware*, que formam a solução que apresentamos: a solução de *software* é uma plataforma

de gestão na forma de uma aplicação *Web*. Os capítulos 3 e 4 descrevem a fase de análise, onde se apresentam o modelo de domínio, os casos de uso e os requisitos não funcionais, e o desenho, onde se referem o diagrama de classes, a arquitetura do sistema e o modelo de dados. O capítulo 5 descreve a implementação da plataforma de gestão e termina com a apresentção dos resultados obtidos.

A solução de *hardware* descreve a constituição de um protótipo e de um produto final. O capítulo 6 aborda a conceção e implementação do protótipo, os resultados obtidos e as razões que nos levaram à implementação de uma solução feita à medida. O produto final, que designamos de Controlador SRI, é apresentado no capítulo 7. Este capítulo inclui o desenho do *hardware*, com uma descrição detalhada dos componentes utilizados e como estes se interligam, e a descrição do *software* embebido no controlador. O capítulo contém ainda uma secção dedicada à poupança energética.

O capítulo 8 introduz aspetos abordados durante o projeto, mas que por limitações várias não foram concluídos e o capítulo 9 faz um balanço geral dos resultados obtidos e apresenta uma introspeção sobre o trabalho desenvolvido.

Finalmente, acresentamos informação adicional, em apêndice, que complementa a compreensão do trabalho desenvolvido.

Capítulo 2

Visão

2.1 Oportunidade de negócio

O projeto SRI pretende auxiliar os profissionais de rega na tarefa de irrigação de jardins públicos oferecendo uma forma de programar remotamente os componentes de rega. A necessidade de desenvolver um sistema deste tipo nasceu após a identificação dos seguintes problemas:

1. a atualização do plano de rega tem de ser feita *in loco* e individualmente para cada controlador na maior parte das instalações em zonas verdes;

2. muitos sistemas não se adequam automaticamente às alterações climatéricas;

3. os sistemas que utilizam comunicação sem fios obrigam à montagem das infrastruturas nos jardins (e.g., antenas, repetidores) e requerem um ponto de alimentação (220V), bem como um ponto de acesso à Internet;

4. os sistemas de rega não registam o consumo de água.

2.2 Alternativas e produtos concorrentes

Nesta secção, identificamos os produtos com características semelhantes existentes no mercado. Nas pesquisas realizadas *online* utilizámos motores de busca generalistas e a base de dados do Gabinete Europeu de Patentes [EPO13]. A procura incidiu sobre sistemas de rega em geral, sistemas de rega com controlo remoto através de *General Packet Radio Service (GPRS)*, tipos de sensores utilizados, técnicas para a obtenção de dados meteorológicos e modelos de controladores de rega, incluindo as suas características e modo de funcionamento.

2.2.1 Estudo de mercado

O estudo de mercado incidiu em produtos que são potenciais concorrentes da nossa solução, destacando-se aqueles que permitem programação remota. Os produtos encontrados, dividem a sua oferta em plataforma de gestão (quando existente), controlador, sensores e restante material necessário à rega (e.g., válvulas solenóides). Na maioria dos casos a plataforma de gestão é específica dos controladores do mesmo fabricante. Neste capítulo, o termo "sistema de rega" denota a plataforma de gestão e os respetivos controladores.

Os sistemas de rega denominados *inteligentes* conseguem fazer a gestão automática da rega baseada num conjunto de parâmetros, como por exemplo, precipitação, humidade do solo, meteorologia, histórico de rega, tipos de plantas e zona geográfica. A forma como esta informação é obtida e processada é uma das principais características diferenciadora destes sistemas de rega. A tabela 2.1 apresenta uma comparação para os produtos que consideramos mais relevantes, diferenciando entre três tipos de gestão da rega automática: via estudo dos solos, recolhendo diversos tipos de informação, via utilização de sensores locais que recolhem informação regularmente ou via obtenção de dados de uma estação meteorológica, normalmente através da Internet [oti12]. A tabela mostra ainda o tipo de interface utilizado nos diferentes sistemas, ou seja, a forma como o operador pode interagir com o sistema programando um plano de rega, obtendo informações dos sensores, etc. Para o tipo de interface diferenciamos também três formas de interação:

1. local no controlador: normalmente através de um pequeno ecrã e botões;
2. remotamente sem um servidor intermediário: comunicação direta, e.g., via rádio através de um dispositvo semelhante a um comando ou de um computador com um adaptador especial;
3. remotamente com utilização de um servidor intermediário: o operador interage com o servidor para definir o planeamento da rega e obter diversas informações.

As características apresentadas nesta tabela mostram uma grande heterogeneidade entre os produtos. Contudo, observa-se que, dos quinze produtos, existem apenas quatro que suportam todas as formas de gestão da rega automática, sendo o controlador SRI um deles. É interessante referir que desses quatro produtos, todos eles suportam o tipo de interação "remotamente com utilização de um servidor intermediário". Quase todos os produtos possibilitam a programação local sendo o controlador SRI neste caso uma exceção à regra pois este não é um requisito do cliente.

A tabela 2.2 compara características mais técnicas relacionadas com o tipo de alimentação e o tipo de protocolo utilizado para comunicação remota.

Finalmente, a tabela 2.3 apresenta uma comparação dos sensores utilizados para os mesmos sistemas de rega apresentados anteriormente. Como se pode observar, apenas

três produtos suportam mais de quatro sensores para monitorização das condições locais. O SRI destaca-se fortemente nesta comparação sendo o único que suporta todos os sensores. Esta vantagem é possível pois o controlador SRI é reprogramável e possui portas de entrada universal com suporte a diferentes tipos de dispositivos (como é o caso dos sensores).

		Gestão da rega automática			**Interface de gestão**		
Marca	**Modelo**	**Via informação de solos, plantas, histórico da zona, etc**	**Via sensores locais**	**Via informação meteorológica**	**Local no controlador**	**Remotamente sem servidor**	**Remotamente com servidor**
RainDrip	WeatherSmart Pro Controller	✗	✗	✓	✓	✗	✗
Rainbird	-	✗	✓	✗	✓	✗	✗
Calsense	ET2000e	✗	✓	✓	✓	✓	✓
Cyber-rain	Long Range Pro Controller	✗	✓	✓	✗	✓	✗
ET Water	Smart Controller, Model 205	✓	✓	✓	✓	✗	✓
Hunter (IMMS)	-	✗	✓	✗	✓	✓	✗
HydroPoint Weather TRAK	WirelessFlow	✓	✓	✓	✓	✗	✓
HydroSaver	ETIC	✗	✓	✗	✓	✓	✓
Irrisoft	-	✗	✓	✓	✓	✗	✓
RainMaster da Irritrol (iCentral)	-	✗	✓	✓	✓	✗	✓
Toro (TriComm System)	TMC-424E	✗	✓	✗	✓	✓	✗
RainBird (IQ Central Controller)	ESP-LXME / ESP-LXD Satellites	✗	✓	✓	✓	✗	✓
Tucor	RKD / RKS	✗	✓	✓	✓	✗	✓
Weathermatic (SmartLink)	Smartline	✓	✓	✓	✓	✗	✓
SRI	**-**	✓	✓	✓	✗	✗	✓

Tabela 2.1: Sistemas de rega e suas características (parte I)

		Alimentação			**Comunicação sem fios**		
Marca	**Modelo**	**≤9V DC**	**24V AC**	**220/120V AC**	**Rádio**	**WI-FI**	**GPRS**
RainDrip	WeatherSmart Pro Controller	✗	✗	✓	✗	✗	✗
Rainbird	-	✓	✗	✗	✗	✗	✗
Calsense	ET2000e	✗	✗	✓	✓	✓	✓
Cyber-rain	Long Range Pro Controller	✗	✗	✓	✗	✓	✗
ET Water	Smart Controller, Model 205	✗	✗	✓	✗	N/A	N/A
Hunter (IMMS)	-	✗	✓	✗	✓	✓	✓
HydroPoint Weather TRAK	WirelessFlow	✗	✓	✗	✗	✗	✓
HydroSaver	ETIC	✗	✗	✓	✓	✗	✗
Irrisoft	-	✗	✓	✗	✗	✓	✗
RainMaster da Irritrol (iCentral)	-	✗	✗	✗	✗	✗	✓
Toro (TriComm System)	TMC-424E	✗	✗	✓	✓	✗	✗
RainBird (IQ Central Controller)	ESP-LXME / ESP-LXD Satellites	✗	✗	✓	✓	✓	✓
Tucor	RKD / RKS	✗	✗	✓	✗	✗	✓
Weathermatic (SmartLink)	Smartline	✗	✓	✗	✓	✗	✓
SRI	**-**	✓	✗	✗	✗	✗	✓

Tabela 2.2: Sistemas de rega e suas características (parte II)

2.2.2 Pesquisa de patentes existentes

Quando se desenvolve um novo produto com foco na comercialização é imprescindível fazer uma pesquisa prévia para encontrar produtos já existentes cujas características tenham sido patenteadas, indepedentemente destes estarem a ser comercializados. A pesquisa que fizemos foca-se nas seguintes características:

- controlo remoto de sistemas de rega;
- sistemas de rega de baixo consumo;
- sistemas inteligentes de rega;
- programação remota de dispositivos.

		Sensores								
Marca	**Modelo**	**Chuva**	**Evapotranspiração**	**Fluxo**	**Vento**	**Geada**	**Luminosidade**	**Humidade**	**Temperatura**	**Hum. do solo**
RainDrip	WeatherSmart Pro Controller	-	-	-	-	-	-	-	-	-
Rainbird	-	-	-	-	-	-	-	-	-	-
Calsense	ET2000e	✓	✓	✓	✓	-	-	-	-	✓
Cyber-rain	Long Range Pro Controller	✓	✗	✓	✗	✗	✗	✗	✗	✗
ET Water	Smart Controller, Model 205	✓	✓	✓	✗	✗	✗	✗	✗	✗
Hunter (IMMS)	-	✓	✓	✓	✓	✓	✗	✗	✗	✗
HydroPoint Weather TRAK	WirelessFlow	✓	✓	✓	✗	✓	✗	✗	✗	✗
HydroSaver	ETIC	✓	✓	✓	✓	✗	✓	✓	✓	✓
Irrisoft	-	✓	✗	✓	✗	✗	✗	✗	✗	✗
RainMaster da Irritrol (iCentral)	-	✓	✗	✓	✗	✓	✗	✗	✗	✓
Toro (TriComm System)	TMC-424E	✗	✗	✓	✗	✗	✗	✗	✗	✗
RainBird (IQ Central Controller)	ESP-LXME / ESP-LXD Satellites	✗	✗	✓	✗	✗	✗	✗	✗	✗
Tucor	RKD / RKS	✓	✓	✓	✗	✗	✗	✗	✗	✓
Weathermatic (SmartLink)	Smartline	✓	✗	✓	✗	✓	✗	✗	✗	✗
SRI	**-**	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabela 2.3: Sistemas de rega e sensores suportados

A tabela 2.4 apresenta os produtos mais relevantes encontrados em pesquisas efetuadas à base de dados do Gabinete Europeu de Patentes (GEP) [EPO13] apresentando para cada um a sua referência e nome.

Uma análise atenta destes documentos permitiu-nos concluir que há uma diferença acentuada entre a descrição do produto que é patenteado e entre e aquilo que na realidade é comercializado. Isto acontece pois no processo de escrita de patente de um produto são descritas inúmeras características que se assemelham à ideia principal para minimizar a concorrência. Por esta razão, à primeira vista, muitos dos resultados da nossa pesquisa parecem idênticos mas na verdade existem algumas subtilezas que os diferenciam. O controlador SRI não é exceção e na secção 2.3 apresentamos algumas características que

Referência	**Nome**
WO2010051652	Wireless irrigation automation and control system
CN1951170	Remote controlled automated irrigation system based on public communication network and control method thereof
ES2176112	Automatic, low cost system to remotely manage and control extensive irrigation networks
US4626984	Remote computer control for irrigation systems
WO0122177	Irrigation control system
EP1008293	Self-contained ecological watering system
WO8704275	Remote process control apparatus
US2012036091	System and method for automated, range-based irrigation
US20130085619	Methods and systems for remote controlling of irrigation systems
US20120290139	Remote control irrigation system
EP2201834	Centralised, automated and remote watering system
WO9727733	Irrigation control system
US7058478	Irrigation controller water management with temperature budgeting
US6898467	Distributed control network for irrigation management

Tabela 2.4: Patentes mais relevantes resultantes da pesquisa à base de dados do GEP

o tornam único e competitivo no mercado.

2.2.3 Estudo de caso - Controlador *RainBird*

O contacto com o cliente permitiu-nos conhecer em detalhe a solução *RainBird*. Apesar desta solução não ser das mais avançadas tecnologicamente e não permitir programação remota, tem uma larga utilização na rega de jardins urbanos atualmente. O estudo desta solução é fundamental quer para identificarmos requisitos funcionais, quer para nos apercebermos das lacunas que devem ser colmatadas no novo produto a desenvolver.

O controlador *RainBird* em estudo é o modelo WP 6 *Series* [Rai13b], que permite gerir até seis válvulas solenóides alimentadas com um impulso de 9V, ao qual se pode ligar um sensor de chuva. Este controlador só pode ser programado localmente com auxílio de um pequeno LCD. Para além da programação básica através do agendamento de tarefas, permite ainda algumas funcionalidades inteligentes, como por exemplo, suspender a rega em caso de chuva (se ligado ao sensor de chuva).

Definição de um ciclo de rega

Para definir um ciclo de rega no controlador *RainBird* é necessário seguir os seguintes passos:

1. Acertar a hora e dia da semana;

2. Definir entre um e três programas escolhendo quais os dias da semana em que deve regar (e.g., o programa A deve regar às segundas e sextas-feiras);

3. Regular a duração de rega *por estação de rega* e atribuição de *cada estação a um programa*;

4. Definir as horas de arranque para cada programa de rega configurado no passo 2 (até um máximo de 8 horas de arranque).

Programa	O que faz o programa?	Horas de Arranque (máx. 8)
A	Rega 2as e 6as	6h, 20h
B	Rega 3as e 5as	5h, 8h, 22h
C	Nunca rega	-

Tabela 2.5: Exemplo da definição de três programas distintos

Estação	Tempo de rega da estação	Programa escolhido para a estação
1	20 min	A
2	13 min	B
3	30 min	C

Tabela 2.6: Exemplo da definição de um ciclo de rega com base nos programas previamente definidos

Descrição do plano definido nas tabelas 2.5 e 2.6

- Na estação 1, a rega será ativada à 2a e 6a (programa A) durante 20 minutos, 2 vezes ao dia nas horas definidas;
- Na estação 2, a rega será ativada à 3a e 5a (programa B) durante 13 minutos, 3 vezes ao dia nas horas definidas;
- Na estação 3, a rega nunca será ativada (programa C está vazio).

Limitações

No controlador *RainBird* existem várias limitações que se prendem sobretudo com o conceito de programa e com o tempo de rega das estações.

- Está limitado a um número máximo de 3 programas;
- Mesma hora de arranque para cada programa. Se o programa A rega às segundas e sextas-feiras e tem como horas de arranque “6h, 20h”, então todas as ações de rega de segunda e sexta desse programa têm a mesma hora de arranque;

- Há um tempo de rega específico para cada estação;

- Uma estação tem no máximo um programa associado e portanto tem inevitavelmente as limitações desse programa.

2.3 Características diferenciadoras

As secções anteriores mostram que, apesar de alguns sistemas já existentes no mercado permitirem a rega remota, o SRI é uma oportunidade de negócio viável e prometedora. De seguida apresentamos as principais vantagens que este projeto acrescenta.

Compatibilidade A arquitetura é versátil em relação aos controladores que podem ser instalados. A programação por módulos permite facilmente a adaptação de vários tipos de controladores e de sensores, facilitando a evolução do sistema com o surgimento de novas tecnologias. A plataforma central é independente dos controladores desde que estes obedeçam ao protocolo de comunicação.

Por outro lado, fisicamente, o controlador desenvolvido pretende ser o mais compatível possível com os sistemas já instalados. Este cuidado prende-se com a escolha da alimentação energética e pelos modelos de válvulas que podem ser utilizados.

Mercado Muitos dos sistemas de rega utilizados na Europa (nomeadamente em Portugal) são produtos provenientes dos Estados Unidos e muitas vezes com programações específicas desse país. Um produto específico para o mercado Europeu pode resultar na exploração de um nicho de mercado mais reduzido e com mais sucesso.

Autonomia energética Os controladores que encontrámos e que comunicam através de GPRS não estão preparados para funcionar com uma bateria. O SRI foi pensado especialmente para permitir a utilização de controladores autónomos energeticamente por um longo período de tempo, estendendo a periodicidade com que é preciso fazer a manutenção do sistema.

Programação da rega O SRI introduz um novo paradigma baseado no tipo de rega de cada válvula. Esta solução permite a gestão de centenas de jardins de uma só vez, libertando o profissional de jardinagem para aquilo que é realmente importante: a manutenção física do jardim. Outra característica importante é a utilização de um calendário na programação de um plano de rega. Esta visão tem uma curva de aprendizagem curta e elimina a necessidade de definir um tempo de rega específico para cada estação.

2.4 Mercado alvo

A identificação do mercado alvo é uma tarefa crucial quando se quer comercializar um produto. Com base nos conhecimentos do cliente que nos foram transmitidos e numa introspeção sobre aplicações do nosso produto, identificámos um conjunto de potenciais compradores do SRI. Em baixo enumeramos por ordem descrescente de importância os compradores mais relevantes identificados:

1. Câmaras municipais;
2. Empresas privadas com propriedades orientadas ao desporto (e.g., campos de golfe, futebol);
3. Empresas privadas com propriedades orientadas à agricultura;

Capítulo 3

Análise

Após as primeiras reuniões e o levantamento do estado da arte, foi elaborada a primeira versão do *Documento de Especificação*. Este documento incluiu o levantamento dos requistos que são apresentados neste capítulo. A análise que aqui apresentamos inclui o modelo de domínio, a descrição textual dos casos de uso e os requisitos não funcionais.

3.1 Modelo de domínio

O Modelo de domínio pretende ilustrar as principais entidades que constituem a aplicação e a forma como estão relacionadas. A figura 3.1 apresenta o Modelo de Domínio para a aplicação SRI.

3.1.1 Descrição das entidades

Todas as entidades têm uma referência diferente. O objetivo deste campo é ter uma forma de identificação de cada entidade independente da implementação (e.g., chave primária na base de dados). Embora os utilizadores tenham conhecimento desta referência, ela é sempre gerada automaticamente na criação de uma entidade.

Jardim Um jardim é um espaço físico onde podem ser instalados vários controladores. A informação do jardim é composta pela referência, nome, comentários e coordenadas. O sistema guarda um catálogo de jardins.

Controlador Um controlador pertence a um único jardim. A sua informação é composta pela referência, nome, comentários e código de barras. O código de barras é um número único de cada controlador e deve coincidir com o número de identificação do controlador real. Para adicionar um controlador, é necessário escolher primeiro qual o jardim onde o controlador está instalado, ou seja, não é possível ter controladores que não estejam associados a um jardim. Para além da ligação jardim-controlador, o sistema deve

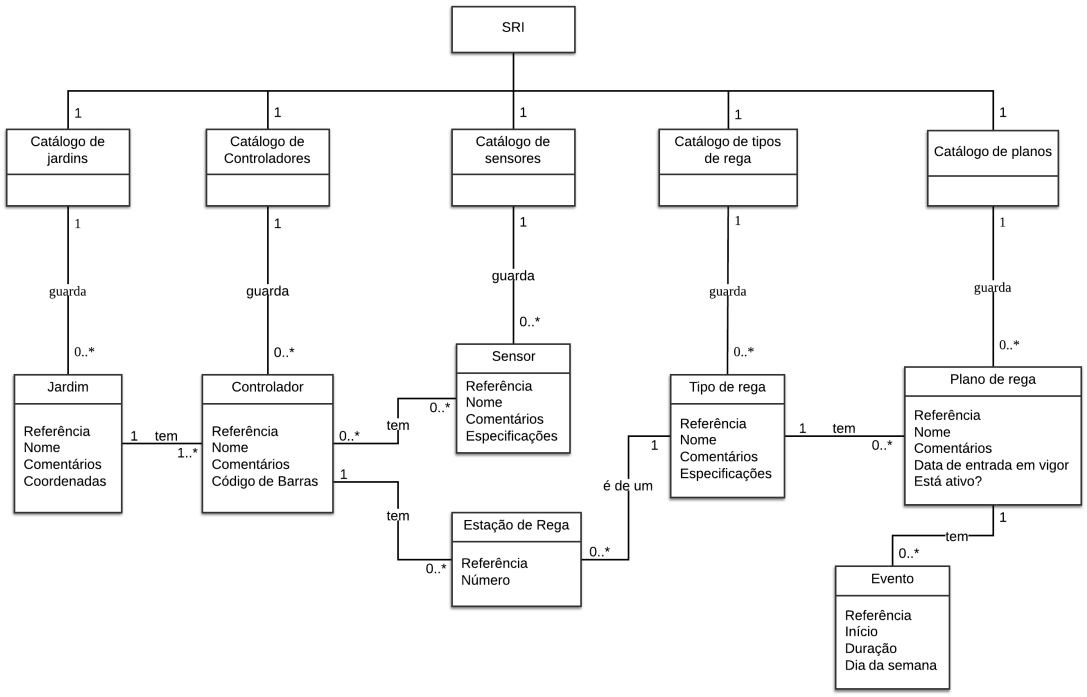


Figura 3.1: Modelo de Domínio para a aplicação SRI

ter um catálogo de controladores que gere todas as operações relacionadas com a rega (e.g., ativar uma válvula a uma determinada hora, ler um sensor, etc.).

Sensor Cada controlador pode ter vários sensores (não é obrigatório) com características distintas. No entanto, vários controladores podem partilhar o mesmo tipo de sensor (e.g., dois controladores diferentes têm ambos um sensor de humidade do solo cada um). Assim, o sistema deve possuir um catálogo de sensores que após serem criados poderão ser associados aos respetivos controladores. O sensor tem um carácter dinâmico no sentido em que pode ser adicionado e removido em qualquer altura de um determinado controlador.

Estação de rega A estação de rega é consituída na realidade por uma válvula solenóide. A função da estação é ativar ou desativar a irrigação de uma zona do jardim. Para adicionar uma estação, é necessário escolher primeiro qual o controlador onde está instalada, ou seja, não é possível ter estações de rega que não estejam associados a um controlador. Para além da referência, cada estação tem um número associado que a identifica no controlador e que tem uma relação direta com a instalação física (i.e., a porta do controlador onde a válvula está ligada). Dependendo da zona onde a válvula está localizada, esta tem sempre um tipo de rega associado.

Tipo de rega O tipo de rega define as características da irrigação para uma determinada zona do jardim. O tipo de rega varia em função do tipo de solo, da exposição solar, das plantas predominantes, etc. Uma vez que existem vários jardins com zonas que têm o mesmo tipo de rega, existe no sistema um catálogo de tipos de rega que poderão ser associados a uma determinada estação de rega de um controlador. O conceito de tipo de rega é muito importante pois torna fácil a programação da rega mesmo quando se gere uma grande quantidade de jardins. Embora possam existir centenas de estações, existem relativamente poucos tipos de rega. Como uma estação de rega tem sempre um tipo de rega associado, basta definir um plano de rega para cada tipo e a programação está feita.

Plano de rega O plano de rega é composto por uma referência, nome, comentários, data de entrada em vigor, um campo que define se o plano está ativo e um conjunto de eventos. Um plano está ativo se estiver a ser executado por algum controlador e está associado obrigatoriamente a um tipo de rega mesmo quando inativo. Um plano pode estar associado indiretamente a uma ou mais válvulas a partir do tipo de rega (e.g., um plano que está associado ao tipo de rega X que por sua vez está associado a dez estações de rega).

Evento Cada plano tem um conjunto de eventos que definem os momentos de execução da rega numa semana. Um evento é constituído por uma referência, dia da semana, início (hora e minutos), duração (hora e minutos).

3.2 Casos de uso

A captação dos objetivos de um projeto exige um acordo de requisitos do novo sistema entre o analista e os clientes. Como forma de descrever cada finalidade do sistema, recorre-se à descrição textual de casos de uso. Os casos de uso são exemplos de utilização do sistema e cobrem os vários cenários de utilização. Um caso de uso pressupõe um ator principal que realiza ações e o sistema a quem são delegadas ações.

Pode portanto dizer-se que os casos de uso têm duas funções fundamentais:

- representam um contrato entre as partes interessadas no sistema;
- são uma referência para o programador.

Os casos de uso que escrevemos dizem respeito às operações executadas pelo utilizador final e que a nossa aplicação deve suportar. Estas operações foram identificadas durante as diversas reuniões com os intervenientes no projeto e são independentes do tipo de tecnologia utilizada (e.g., *browser Web*, aplicação móvel, aplicação *desktop*, etc.). Neste capítulo descrevemos apenas alguns casos de uso que dão a noção das operações mais comuns (e.g., pesquisar entidade, adicionar entidade, etc.) suportadas pela aplicação. A

tabela 3.1 enumera todos os casos de uso identificados cuja descrição completa pode ser consultada em anexo na secção A.

Grupo	**Código**	**Nome do caso de uso**
Jardins	CU-J1	Pesquisar jardins
	CU-J2	Gerir jardim
	CU-J3	Adicionar jardim
	CU-J4	Editar jardim
	CU-J5	Remover jardim
Planos de rega	CU-PR1	Gerir plano de rega
	CU-PR2	Adicionar plano de rega
	CU-PR3	Editar plano de rega
	CU-PR4	Remover plano de rega
	CU-PR5	Adicionar evento ao plano de rega
	CU-PR6	Remover evento do plano de rega
	CU-PR7	Definir como plano de rega em vigor
Tipos de rega	CU-PR1	Gerir tipo de rega
	CU-TR2	Adicionar tipo de rega
	CU-TR3	Editar tipo de rega
	CU-TR4	Consultar planos de rega associados
Controlador	CU-C1	Gerir controlador
	CU-C2	Adicionar controlador
	CU-C3	Editar controlador
	CU-C4	Remover controlador
Estações de rega	CU-ER1	Adicionar estação de rega
	CU-ER2	Editar estação de rega
	CU-ER3	Remover estações de rega
	CU-ER4	Parar a rega imediatamente
	CU-ER5	Iniciar a rega imediatamente
	CU-ER6	Desativar a rega temporariamente
Sensores	CU-S1	Adicionar sensor
	CU-S2	Editar sensor
	CU-S3	Remover sensores
	CU-S4	Consultr registos do sensor

Tabela 3.1: Casos de uso SRI

3.2.1 Pesquisar entidade

Existem cenários em que a grande quantidade de um mesmo tipo de entidade pode justificar a existência de uma pesquisa. É o caso da entidade "Jardim" que não pertence a nenhuma outra entidade e que torna mais complexa a sua identificação e seleção. A

pesquisa por jardins deve possuir um conjunto de filtros que permite encontrar mais facilmente um jardim ou conjunto de jardins específicos. Por exemplo, pesquisar todos os jardins de uma freguesia.

Caso de Uso: Pesquisar jardins.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O sistema devolve a lista de jardins que satisfazem os critérios de pesquisa.
Cenário principal de Sucesso:

1. O operador indica que pretende fazer uma pesquisa de jardins;
2. O sistema devolve uma lista com nomes de filtros para pesquisar jardins;
3. O operador seleciona um filtro;
4. O sistema apresenta um formulário com os campos a preencher para o filtro selecionado;
5. O operador completa as informações para cada filtro;
6. O operador escolhe a opção "Pesquisar";
7. O sistema devolve a lista dos jardins definidos no sistema contendo os atributos "referência", "nome", "latitude", "longitude" e "estado" com base no filtro selecionado;
8. O operador indica que quer ver mais jardins;
9. O operador repete os passos 7 e 8 enquanto desejar ou até o sistema avisar que não há mais jardins para mostrar.

Extensões:
7A - O sistema mostra a mensagem "Não existem resultados para o filtro selecionado" e retoma-se o cenário principal a partir do passo 2
Observações: O estado do jardim é ativo se pelo menos uma das estações de um dos controladores estiver a regar ou é inativo se nenhuma estação estiver a regar.

3.2.2 Adicionar entidade

Uma das operações fundamentais é adicionar uma nova entidade. Todos os grupos de casos de uso que escrevemos possuem esta operação. Regra geral, um tipo de entidade é

adicionado a partir de outro tipo de entidade. Os dois casos de uso que mostramos aqui são exemplo disso. No caso da entidade "Evento de rega", esta só pode ser adicionada se já existir uma entidade "Plano de rega" criada e que esteja a ser gerida. Analogamente, a entidade "Tipo de rega" é adicionada a uma entidade "Controlador". Exceção a esta regra é o caso da entidade "Jardim" (ver secção A.0.1 em anexo). Uma entidade só é adicionada ao sistema após boa validação dos dados submetidos.

Caso de Uso: Adicionar evento ao plano.
Ator principal: Operador.
Pré-condições: Existe um plano de rega em gestão.
Pós-condições: É adicionado um evento ao plano de rega em gestão.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar um evento ao plano de rega;
2. O sistema apresenta um formulário contendo os atributos "dia da semana", "hora de início" e "duração";
3. O operador fornece a informação e escolhe a opção "Adicionar";
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção "Sim";
6. O sistema valida os dados submetidos e mostra uma mensagem a informar que o evento foi adicionado com sucesso.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
5A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e retorna-se ao passo 2.

Caso de Uso: Adicionar estação de rega.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições:
A estação de rega indicada é adicionada ao sistema.
É feita uma associação entre a nova estação e o controlador selecionado.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar uma estação de rega ao controlador;

2. O sistema apresenta um formulário com os atributos "número da porta de instalação da estação no controlador" e uma lista de tipos de rega;
3. O operador fornece a informação e escolhe a opção "Adicionar";
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção "Sim";
6. O sistema valida os dados submetidos, mostra uma mensagem a informar que a estação de rega foi adicionada com sucesso e o caso de uso termina.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
5A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 2.

3.2.3 Gerir entidade

As entidades cuja lógica de negócio representa maior complexidade têm o caso de uso que permite fazer a gestão da entidade. Estas são entidades que tipicamente se relacionam com vários tipos de entidades diferentes como é o caso da entidade "Controlador" que mostramos aqui. Gerir uma entidade não é apenas consultar as suas informações. Tipicamente, ao gerir uma entidade, é possível consultar as suas informações mas também as informações das entidades relacionadas e modificar esses dados. É igualmente a partir deste caso de uso que são executados casos de uso de operações específicas da entidade (como exemplo ver a secção 3.2.5).

Caso de Uso: Gerir controlador.
Ator principal: Operador.
Pré-condições: Existe um jardim em gestão.
Pós-condições: O sistema mostra as informações do controlador indicado.
Cenário principal de Sucesso:

1. O operador seleciona o controlador que pretende gerir;
2. O sistema devolve as informações do controlador contendo os atributos "nome", "referência", "comentários", "código de barras", "nome do jardim onde está instalado", uma lista de eventos contendo os atributos "referência", "hora de início", "hora de fim", "dia da semana" e "válvula", uma lista de estações de rega contendo os atributos "referência", "tipo de rega"e "estado"e uma lista de sensores contendo os atributos "referência", "nome" e "valor lido" e o caso de uso termina;

Extensões:
2A - O sistema mostra a mensagem “Não existem eventos associados ao controlador indicado”.
2B - O sistema mostra a mensagem “Não existem estações de rega associadas ao controlador indicado”.
2C - O sistema mostra a mensagem “Não existem sensores associados ao controlador indicado”.

3.2.4 Consultar informações de uma entidade

Esta categoria de casos de uso descreve operações que permitem obter os dados associados a uma entidade específica. À semelhança do que foi referido na secção 3.2.2, as informações de um tipo de entidade são obtidas a partir de outro tipo de entidade. Os dois casos de uso que apresentamos de seguida são um bom exemplo desta condição. O caso de uso “Consultar planos de rega associados” permite obter dados da entidade “Plano de rega” mas pertence à entidade “Tipo de rega”. Estas relações são inerentes à lógica de negócio, observável no modelo de domínio descrito na secção 3.1. Um cenário idêntico apresenta-se para o caso de uso “Consultar registos do sensor”.

Caso de Uso: Consultar planos de rega associados.
Ator principal: Operador.
Pré-condições: Existe um tipo de rega em gestão.
Pós-condições: O sistema devolve uma lista com os planos de rega associados ao tipo de rega em gestão.
Cenário principal de Sucesso:

1. O operador indica que pretende consultar os planos de rega associados;
2. O sistema devolve uma lista de planos de rega associados ao tipo de rega em gestão e o caso de uso termina;

Extensões:
2A - O sistema mostra a mensagem “Não existem planos de rega associados” e o caso de uso termina.
3A - O operador escolhe a opção “Cancelar” e o caso de uso termina.

Caso de Uso: Consultar registos do sensor.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão com pelo menos um sensor adicionado.

Pós-condições: O sistema apresenta uma lista de registos com as leituras dos sensores selecionados.
Cenário principal de Sucesso:

1. O operador seleciona um ou mais sensores;
2. O operador indica que pretende consultar os registos dos sensores;
3. O sistema devolve uma lista de registos com as leituras dos sensores selecionados e o caso de uso termina.

Extensões:
3A - Não existem registos para os sensores selecionados e o caso de uso termina.

3.2.5 Outras operações

Finalmente incluímos dois casos de uso que dizem respeito a operações exclusivas de cada tipo de entidade: o caso de uso "Definir como plano de rega em vigor" que apenas faz sentido para a entidade "Plano de rega" e o caso de uso "Iniciar a rega imediatamente" que pertence à entidade "Estação de rega".

Caso de Uso: Definir como plano de rega em vigor.
Ator principal: Operador.
Pré-condições:
Pós-condições: O plano indicado passa a ser o plano ativo para o tipo de rega indicado.
Cenário principal de Sucesso:

1. O operador indica que pretende definir um plano de rega como o plano ativo para o seu tipo de rega associado;
2. O sistema mostra um formulário contendo o campo "data de entrada em vigor";
3. O operador fornece a informação e escolhe a opção "Definir como plano de rega em vigor";
4. O sistema mostrar uma mensagem a informar que o plano de rega irá ficar em vigor a partir da data indicada no passo 3.

Extensões:
4A - O sistema mostra a mensagem de erro "a data não é válida" a informar que o plano de rega não pode ficar em vigor.
4B - O sistema mostra a mensagem de erro "já existe outro plano agendado para a mesma data" a informar que o plano de rega não pode ficar em vigor.

Caso de Uso: Iniciar a rega imediatamente.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições: A rega é iniciada nas estações de rega selecionadas.
Cenário principal de Sucesso:

1. O operador seleciona uma ou mais estações de rega;
2. O operador indica que pretende iniciar a rega nas estações de rega selecionadas;
3. O sistema pede para o operador confirmar o pedido;
4. O operador escolhe a opção "Sim";
5. O sistema mostra uma mensagem a informar que a rega foi iniciada nas estações de rega selecionadas e o caso de uso termina;

Extensões:
4A - O utilizador escolhe a opção "Não" e o caso de uso termina.
5A - O sistema mostra a mensagem de erro "só pode iniciar a rega se a estação estiver desativa" e o caso de uso termina.

3.3 Requisitos não funcionais

Os requisitos não funcionais descrevem comportamentos ou características desejáveis na utilização do produto, mas que são independentes das suas funcionalidades. Exemplos comuns deste tipo de requisitos numa aplicação de *software* são velocidade de reposta na interação com o utilizador, segurança da aplicação, tipo de comunicação suportada, etc. De acordo com o cliente, foram identificados os requisitos não funcionais para o SRI presentes na tabela 3.2.

Poupança energética do controlador	O controlador deve ter elevada autonomia energética com duração da bateria igual ou superior a um ano.
Comunicação remota sem fios	O controlador deve ser capaz de comunicar com a plataforma central de forma rápida (na ordem dos segundos) e fiável utilizando uma ligação sem fios e de médio alcance (pelo menos 30 Km).
Usabilidade da aplicação	A plataforma central de gestão deve facilitar a gestão das várias entidades oferecendo uma interface intuitiva e com uma curva de aprendizagem reduzida.
Segurança	A plataforma central deve permitir a autenticação de utilizadores bloqueando certas operações para um conjunto de utilizadores com menos privilégios. Adicionalmente a comunicação com os controladores deve ser segura impendido a utilização inevida dos vários controladores que estão instalados em zonas públicas e que utilizam recursos públicos, como é o caso da água.
Portabilidade e modularidade do software	O *software* implementado tanto na aplicação da plataforma central como no controlador deve seguir uma filosofia que permita a adapatação a diferentes cenários de instalação. No caso da plataforma central a lógica de negócio deve ser independente do tipo de aplicação (e.g., aplicação *Web*, aplicação móvel, etc.). No caso do controlador, o código desenvolvido deve separar bem as responsabilidades (e.g., comunicação móvel, acesso aos sensores, execução do plano de rega, medidas de poupança energética) permitindo uma adaptação fácil a mudanças de *hardware* (e.g., o microcontrolador ou o módulo de comunicação mudam).
Escalabilidade da aplicação	O sistema que gere os controladores deve suportar um nível elevado de escalabilidade. O sistema pode crescer rapidamente devido ao número de jardins adicionados e consequemente ao número de controladores. Uma gestão em massa dos controladores terá inevitavelmente picos de utilização que deverão ser devidamente antecipados.
Time-to-market	A pesquisa do estado da arte revelou diversas soluções existentes no mercado ou patenteadas que podem ser concorrentes ao SRI. Adicionalmente, o nosso cliente tem conhecimento de entidades interessadas num sistema como este colocando uma enorme pressão na comercialização rápida deste produto.

Tabela 3.2: Requisitos não funcionais

Capítulo 4

Desenho e implementação

Após a fase de análise onde são identificados os conceitos e requisitos principais da solução a desenvolver, é importante pensar numa estratégia de *software* que resolva as necessidades e os problemas levantados pelo cliente. Neste capítulo, identificamos os objetos de *software* que devem representar os objetos identificados na fase de análise e como estes colaboram entre si. Para cada um descrevemos os seus atributos e operações de acordo com as suas responsabilidades.

4.1 A utilização do Java Enterprise Edition

A plataforma de gestão que desenvolvemos utiliza a tecnologia Java Enterprise Edition (JavaEE) ideal para o desenvolvimento de aplicações empresariais que são disponibilizadas na *Web*. Uma das grandes vantagens desta tecnologia é a encapsulação da conexão dos vários módulos necessários à implementação de uma aplicação enterpresarial. Assim, é possível, por exemplo, construir uma aplicação *Web* com um domínio robusto, independente, com ligação a uma base de dados e ainda utilizar serviços na *Web* com apenas algumas modificações ao código. A utilização de anotações torna-se especialmente útil, pois permite fazer injeção de dependências, ou seja, ser a própria plataforma JavaEE a criar automaticamente referências entre os vários componentes da aplicação, poupando muito trabalho ao programador. A explicação pormenorizada dos conceitos de JavaEE referidos neste capítulo sai fora do âmbito desta tese. No entanto, uma boa referência é o tutorial oficial [Ora13b].

4.1.1 Aplicação do *Model View Controller*

Um dos padrões de desenho mais utilizados no desenvolvimento de *software* é o *Model View Controller (MVC)* [Fow06]. Este padrão separa conceptualmente uma aplicação em três módulos fundamentais: modelo, vista e controlador. O modelo contém a definição dos objetos que representam as entidades no mundo real. Na nossa aplicação, os controladores (*hardware*) ou os jardins, são exemplos de objetos que constituem o mo-

delo. A definição dos objetos do modelo deve ser independente da forma como estes são apresentados numa determinada interface. Assim, compete ao módulo vista tratar desta apresentação que pode ser através de uma interface gráfica *Web*, uma aplicação *Desktop*, aplicação móvel, etc. Finalmente, o módulo controlador é responsável pela lógica do negócio, ou seja, para uma determinada operação, procura a informação nos objetos do modelo necessários e faz as transformações necessárias. Os resultados são passados à vista e devidamente apresentados. Uma separação de módulos idêntica está bem presente na numa aplicação JavaEE como explicamos nas próximas secções.

4.2 Camada de Negócio

A camada de negócio da plataforma de gestão está concentrada no módulo EJB. Numa aplicação JavaEE, esta camada encaixa-se no módulo controlador do padrão MVC. Contém todo o código responsável pela lógica do negócio e serve de suporte a outras camadas, como por exemplo, a camada *Web*. O módulo EJB do JavaEE acrescenta o conceito de Enterprise Java Beans (EJBs) que são objetos com propriedades especiais e cujo ciclo de vida é gerido automaticamente. Entre estas propriedades destacamos a capacidade de invocação de métodos remotamente (designada de Remote Method Invocation (RMI)), a disponibilização de serviços na *Web* e a utilização de recursos de um sistema de informação empresarial (Enterprise Information System (EIS)), como por exemplo, o acesso a uma base de dados relacional.

O domínio da nossa plataforma contém objetos chave com responsabilidades bem distintas seguindo os padrões de desenho de *software* que melhor se adequam aos requisitos do sistema [Lar01] e que serão descritos mais à frente. Para além da utilização dos objetos EJB referidos, são utilizados também objetos Plain Old Java Object (POJO) cujo ciclo de vida é da responsabilidade do programador. A figura 4.1 pretende ilustrar os intervenientes principais da camada de negócio, e a forma como estes se relacionam entre si, através de uma visão de alto nível. Na figura é possível ver também o papel de cada interveniente na aplicação. De seguida fazemos uma descrição detalhada para cada um destes intervenientes.

4.2.1 Entidades

De acordo com o modelo de domínio descrito na secção 3.1 existem entidades que representam os principais intervenientes no sistema SRI. Estas entidades guardam a informação específica de cada interveniente, expõem as suas operações e mapeiam as interligações entre intervenientes. As entidades são objetos POJO e constituem o módulo modelo do padrão MVC. Cada classe de entidade implementa uma interface como se pode ver o exemplo para a entidade "Jardim" na figura 4.2. As setas a tracejado simbolizam a interface de uma determinada classe. As outras setas representam ligações entre entidades

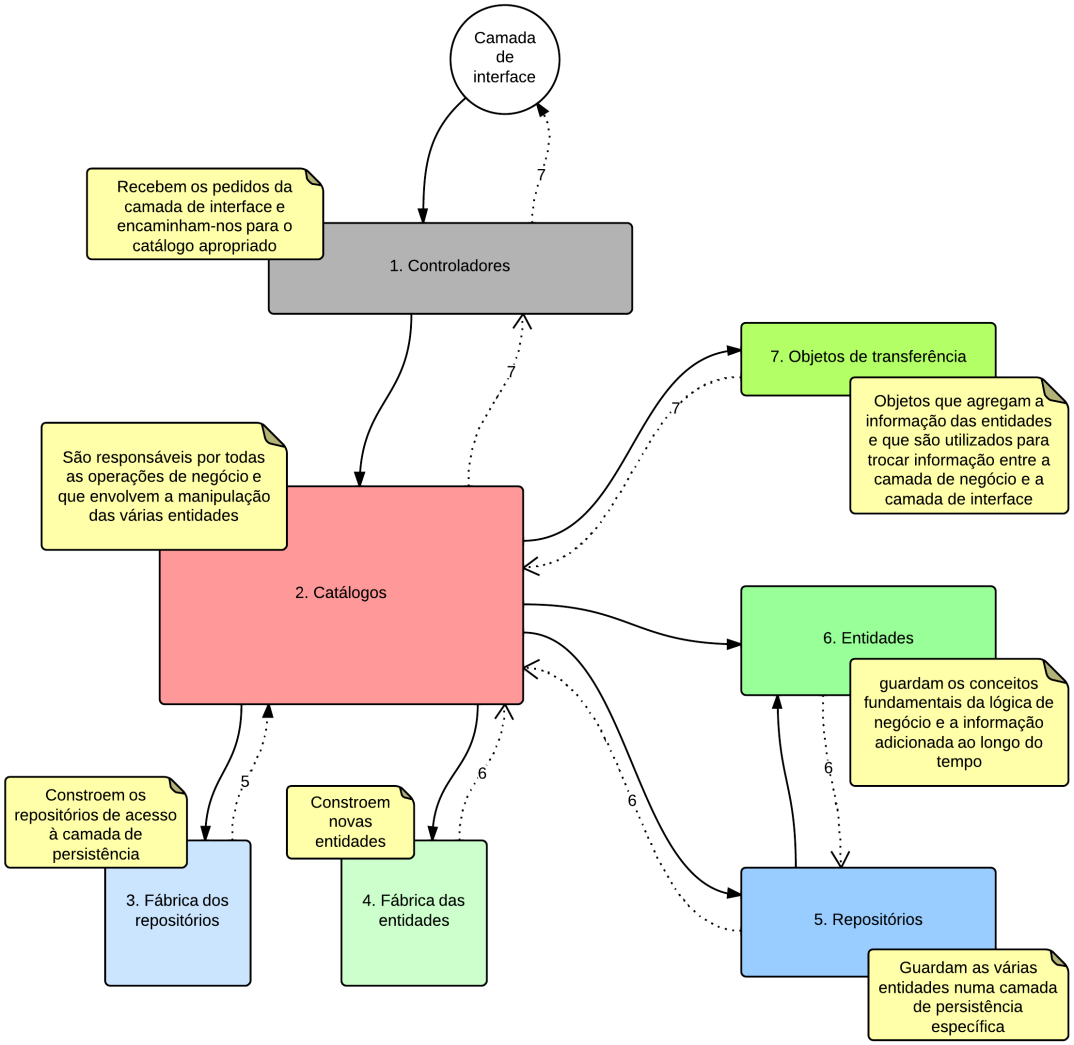


Figura 4.1: Camada de negócio - conceitos principais

relacionadas com os seus atributos (e.g., o jardim tem um atributo que é uma lista de controladores).

Algumas entidades têm uma correspondência direta com o modelo de domínio, como é o caso das entidades “Jardim”, “Controlador”, “Plano de rega”, “Tarefa”, “Tipo de rega” e “Estação de rega”. A entidade “Controlador” é aquela que tem mais ligações com outras entidades. Estas entidades e as suas relações são visíveis nas figuras 4.3 e 4.4. Outras entidades foram adicionadas de forma a enriquecer o modelo de negócio e que descrevemos brevemente em baixo e na figura 4.5:

- “ClientRequest”: contém informação sobre um pedido efetuado na plataforma de gestão e que deve ser convertido e enviado para o controlador. Exemplos de pedidos são “atualizar o plano de rega”, “obter informação dos sensores de rega”, “obter o estado da bateria”, etc. Um pedido é sempre identificado por um código único que

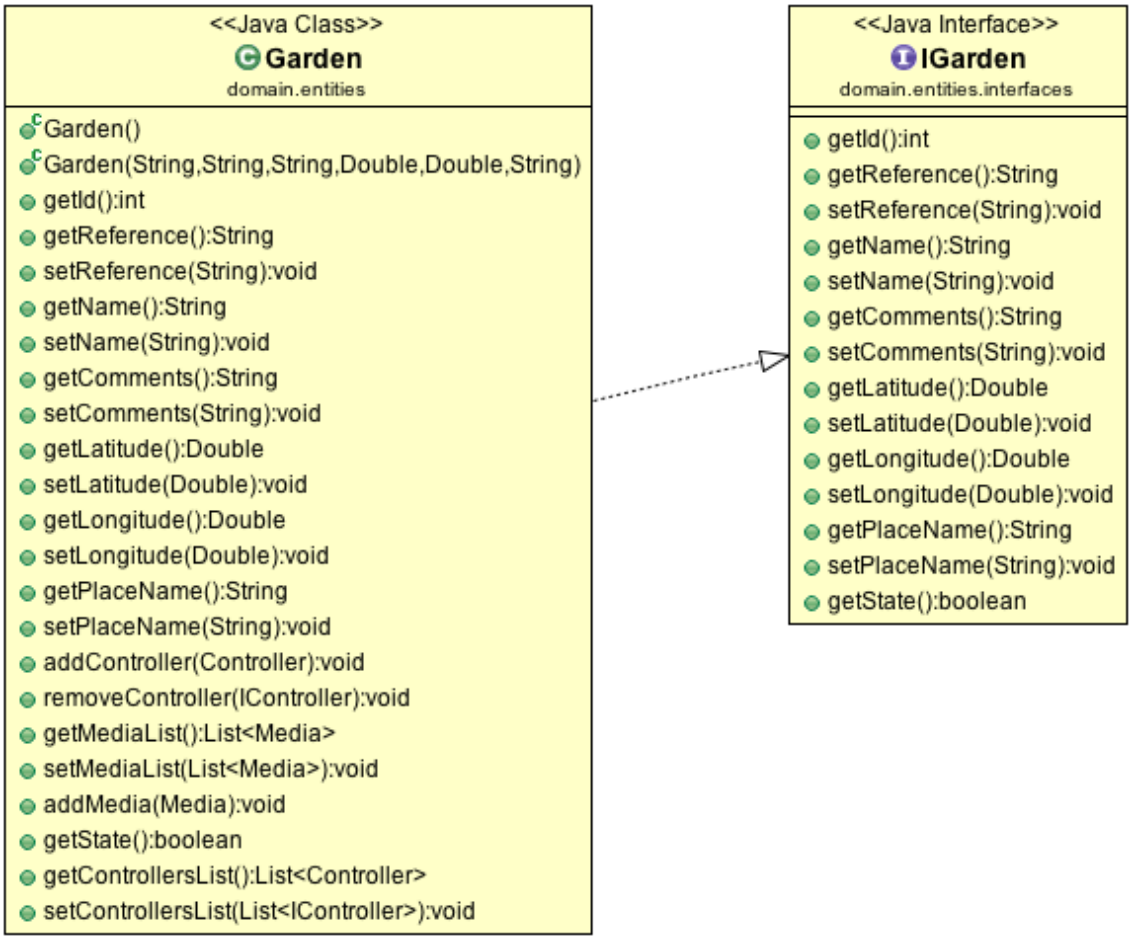


Figura 4.2: Diagrama de classes - exemplo de entidade e sua interface

é conhecido tanto pela plataforma central como pelo controlador;

- "LogEntry": esta entidade auxilia na criação de registos com todas as leituras feitas para cada controlador. Sempre que é iniciada uma nova ligação é criado um novo registo. Cada registo é constituído por várias leituras. É graças a estes registos que será possível ter um histórico para cada controlador;

- "LogReading": representa uma leitura de um sensor específico de um controlador. Cada leitura tem por isso uma associação com um *plugin* (sensor) específico;

- "LogWriting": permite guardar informações relacionadas com os pedidos enviados para o controlador em cada comunicação. Cada instância desta entidade terá informações tais como "todos os pedidos enviados para o controlador foram recebidos e executados com sucesso", "data e duração da comunicação e de cada pedido especificamente", etc;

- "Plugin": esta entidade é a representação lógica de um disposito capaz de ler valores quando ligado a um controlador. Tipicamente, um *plugin* será um sensor relacio-

Figura 4.3: Diagrama de classes - entidades (I)

nado com a rega. A ideia de *plugin* pretende oferecer flexibidade no crescimento do sistema. Uma entidade para cada sensor específico seria mais limitativo se quisermos que o utilizador possa adicionar e remover sensores ao longo do tempo.

As entidades estão na base do modelo de negócio o qual está organizado em função destas. Como veremos nas próximas secções, todos os intervenientes estão ligados direta ou indiretamente com as várias entidades.

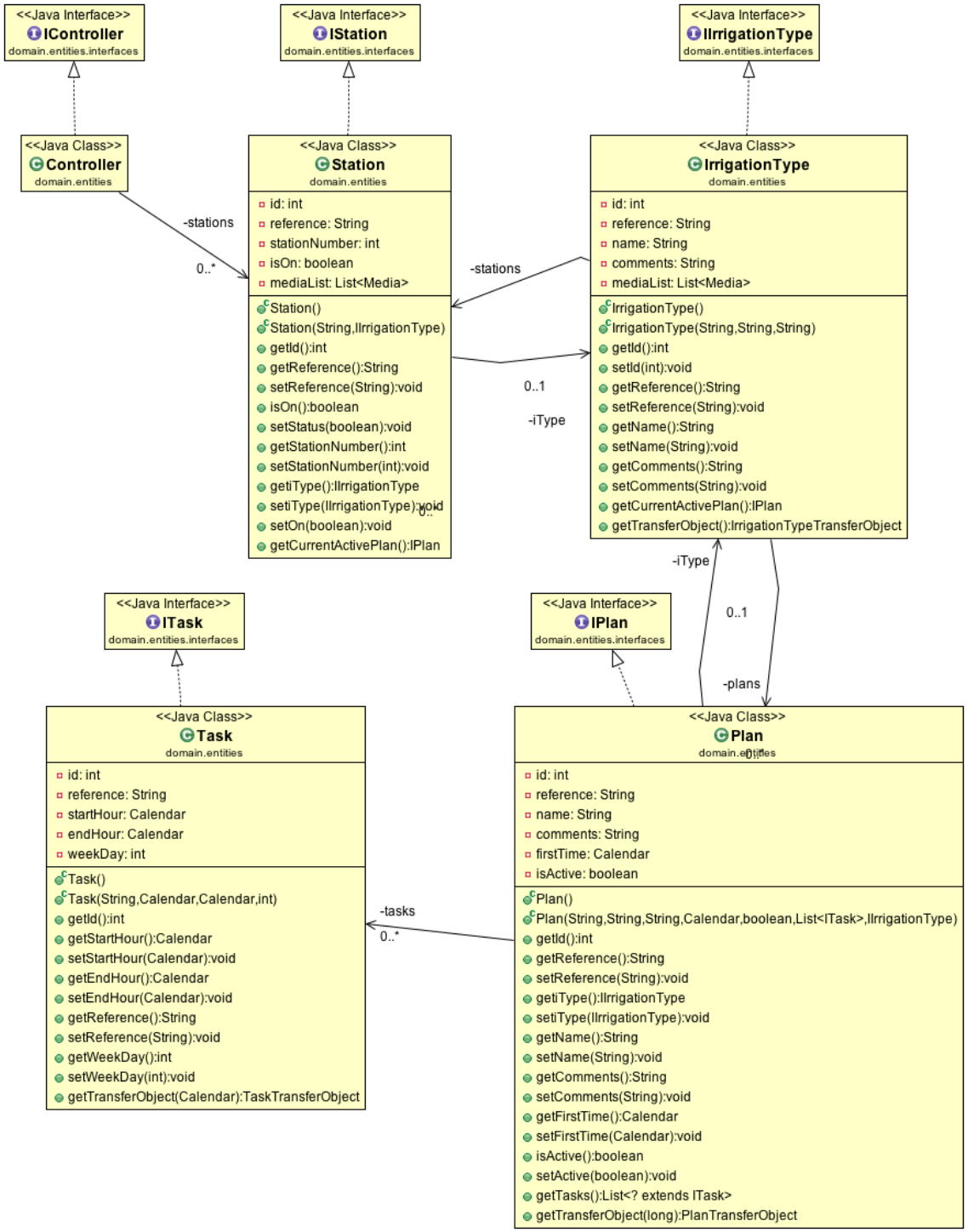


Figura 4.4: Diagrama de classes - entidades (II)

4.2.2 Controladores

Estes objetos representam o padrão de desenho *Controller*, também designado de *Handler* e que lhes dá uma parte do nome. Os controladores são a porta de entrada para os eventos do sistema sendo responsáveis por encaminhar um pedido para o catálogo apropriado. A

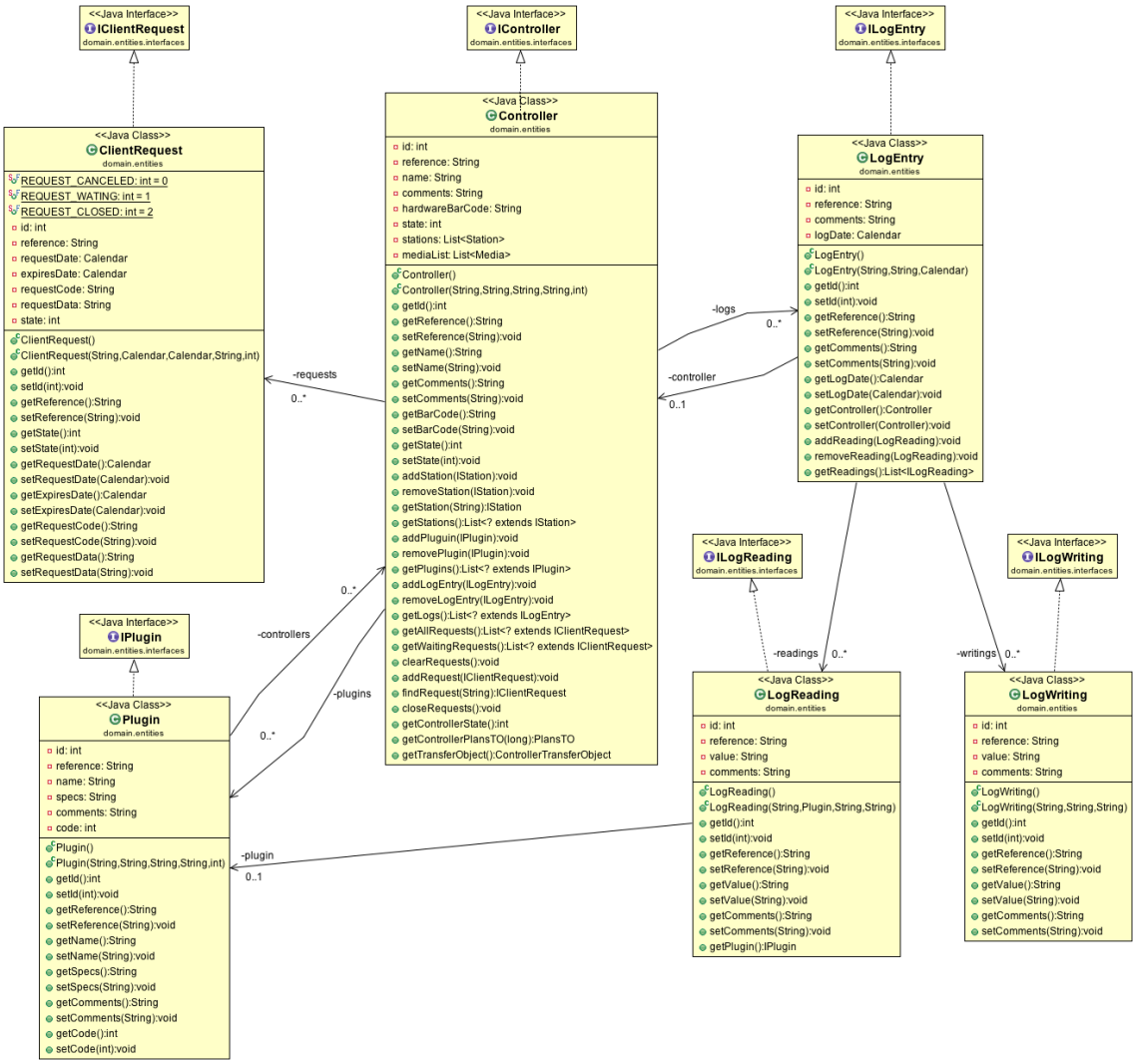


Figura 4.5: Diagrama de classes - entidades (III)

nossa implementação prevê três controladores:

- HandlerGardens (responsável por pedidos relacionados com a manipulação da entidade "Jardim" incluindo a criação de novos controladores de jardins);
- HandlerPlans (responsável por pedidos específicos relacionados com os planos e tipos de rega)
- HandlerControllers (responsável por pedidos de todas as outras entidades e que se relacionam diretamente com a entidade "Controlador")

A figura 4.6 representa os controladores acima referidos.

Todos os controladores são EJBs *stateless*. Os eventos do sistema podem ocorrer de duas formas: através da invocação do controlador a partir do módulo *Web* (ver secção 4.3) ou a através da invocação de um *webservice*.

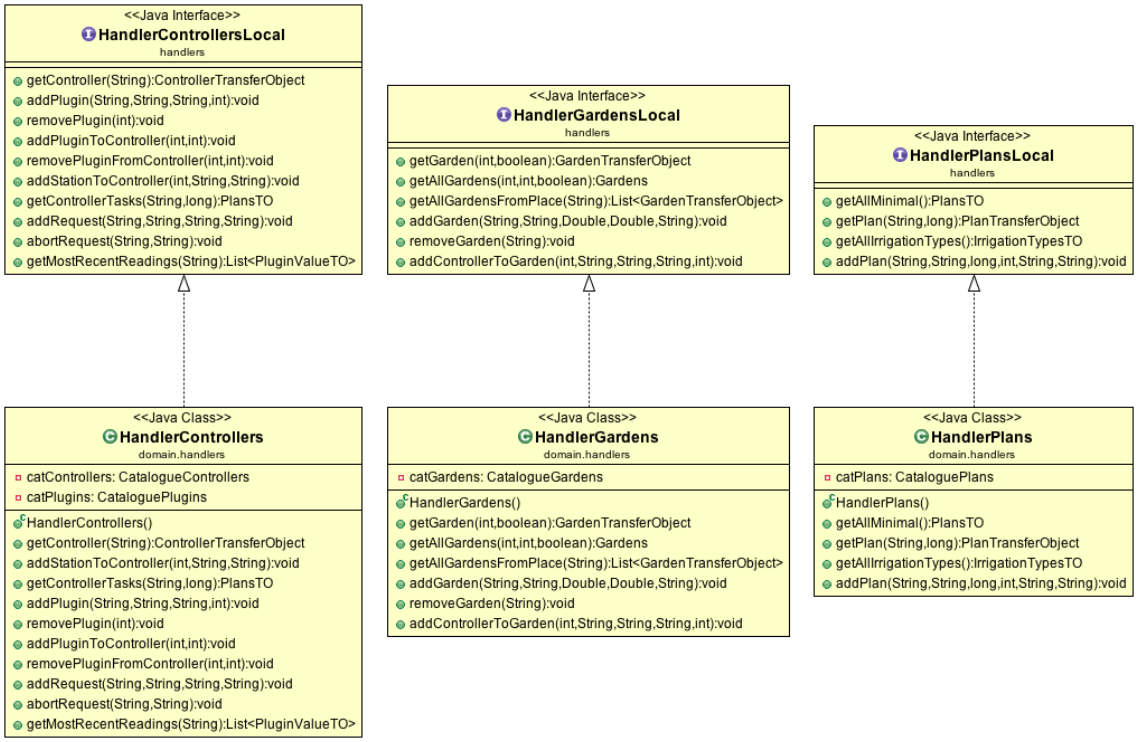


Figura 4.6: Diagrama de classes - controladores

Serviços na *Web*

Os pedidos remotos *Web* que não modificam diretamente o aspeto da página são feitos recorrendo a serviços na *Web*. A plataforma JavaEE facilita a criação destes serviços bastando para isso anotar um EJB com "@Webservice". Os controladores permitem a invocação dos seus métodos através de serviços na *Web*.

4.2.3 Catálogos

Os catálogos guardam a lógica necessária para implementar cada caso de uso específico. Cada catálogo guarda uma referência para o EJB de arranque ("AppLoader") a partir do qual pode aceder às várias fábricas, repositórios e respetivas entidades. Os catálogos implementados seguem a ideia do modelo de domínio com a excepção do cátalogo de tipos de rega cujas operações são neste caso da responsabilidade do catálogo dos planos. A figura 4.7 ilustra todos os catálogos do nosso sistema:"CatalogueGardens", "CatalogueControllers", "CataloguePlans" e "CataloguePlugins". Todos os catálogos são EJBs *stateless*.

4.2.4 Repositórios

Os repositórios encapsulam o acesso à camada de persistência e a sua função principal é expôr um conjunto de operações que permitem guardar e recuperar o estado das várias en-

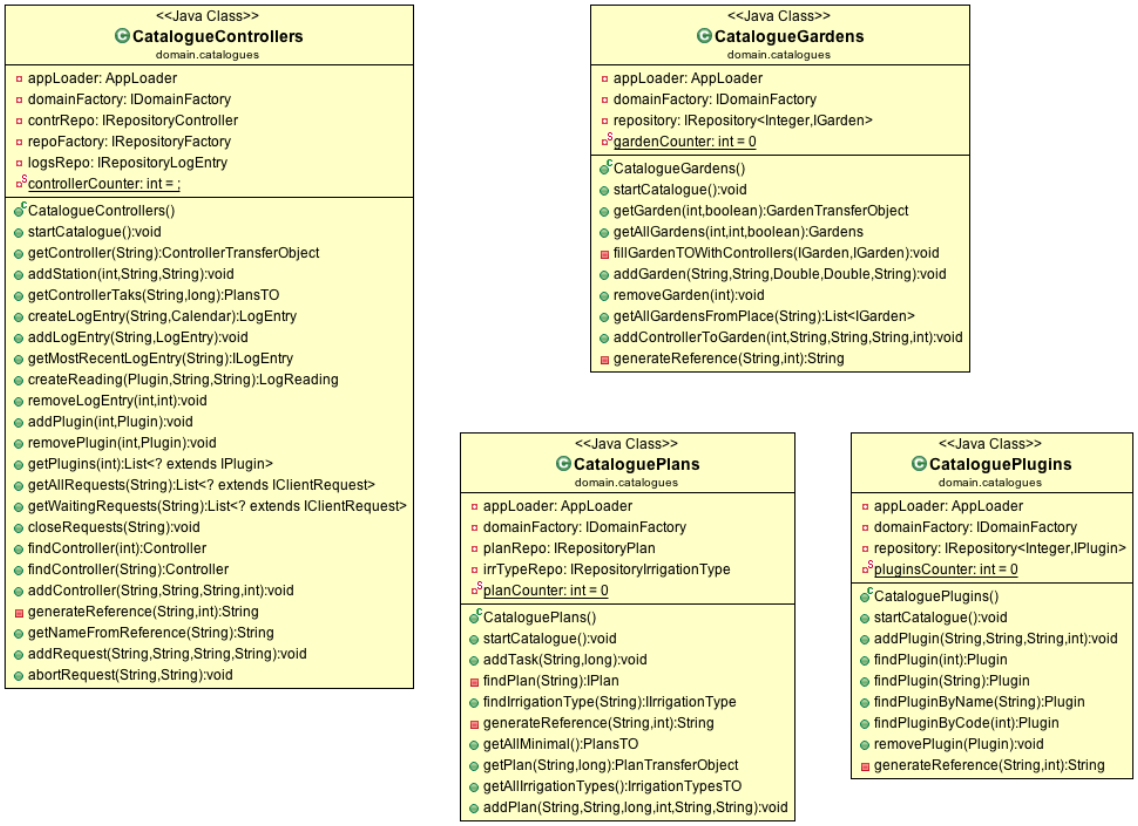


Figura 4.7: Diagrama de classes - catálogos

tidades. A camada de persistência pode variar de acordo com a instalação da aplicação em ambientes diferentes e por isso é importante uniformizar as operações desta camada independentemente do tipo de tecnologia utilizada (e.g., MySQL, Oracle, sistema de ficheiros, memória volátil, etc.). O facto destes repositórios uniformizarem o acesso à camada de persistência faz com que exista um conjunto de operações que são iguais para todas as entidades. Exemplo destas operações são "obter uma instância de uma entidade pela sua referência", "obter todas as instâncias daquela entidade", "remover uma instância da entidade", "guardar uma instância da entidade", etc. Para estas operações, a única diferença é o tipo de entidade que se quer persistir/recuperar. Para satisfazer estas necessidades optámos por ter uma interface (IRepository<K,E>) com um conjunto de operações bem definido e que pode ser implementada de diversas maneiras e para diferentes entidades. A interface "IRepository" possui a característica de ser um tipo genérico. Um tipo genérico possui um ou mais parâmetros de tipo que são mais tarde substituídos quando o tipo é instanciado ou declarado. Esta propriedade permite detetar qual o tipo de entidade que deve ser utilizada em cada declaração da interface evitando ter uma interface diferente para cada entidade. Neste caso concreto, a interface "IRepository" possui dois tipos de parâmetros que dizem respeito ao tipo da chave de pesquisa da entidade (K) e ao tipo da entidade (E), respetivamente.

A versão da aplicação que desenvolvemos e instalámos na fase de desenvolvimento

utiliza uma base de dados MySQL e por isso a nossa implementação para a interface "IRepository" utiliza a tecnologia JPA. A secção 4.2.7 descreve com mais detalhe esta abordagem. Assim, temos uma classe (JPARepository) que implementa os métodos da interface anteriormente referida. Todo o acesso à base de dados é encapsulado por uma inteface do JPA (EntityManager).

Finalmente, pode ainda acontecer que, para além das operações básicas, existem operações que são específicas de cada entidade. Para alcançar esta flexibilidade, a classe "JPARepository" pode ser estendida por outras classes que podem adicionar métodos específicos. É na declaração destas classes que são especificados os tipos de parâmetro.

Para clarificar estes conceitos ilustramos na figura 4.8 o diagrama de classes que exemplifica a utiliação dos intervenientes acima referidos para as entidades "Controller" e "LogEntry". Os repositórios são utilizados pelos vários catálogos referidos na secção 4.2.3 e neste caso concreto incluimos o catálogo "CatalogueControllers" que tem os respetivos repositórios para as duas entidades referidas. A classe "JPARepositoryLogEntry" é um exemplo em que existe um método específico para a respetiva entidade.

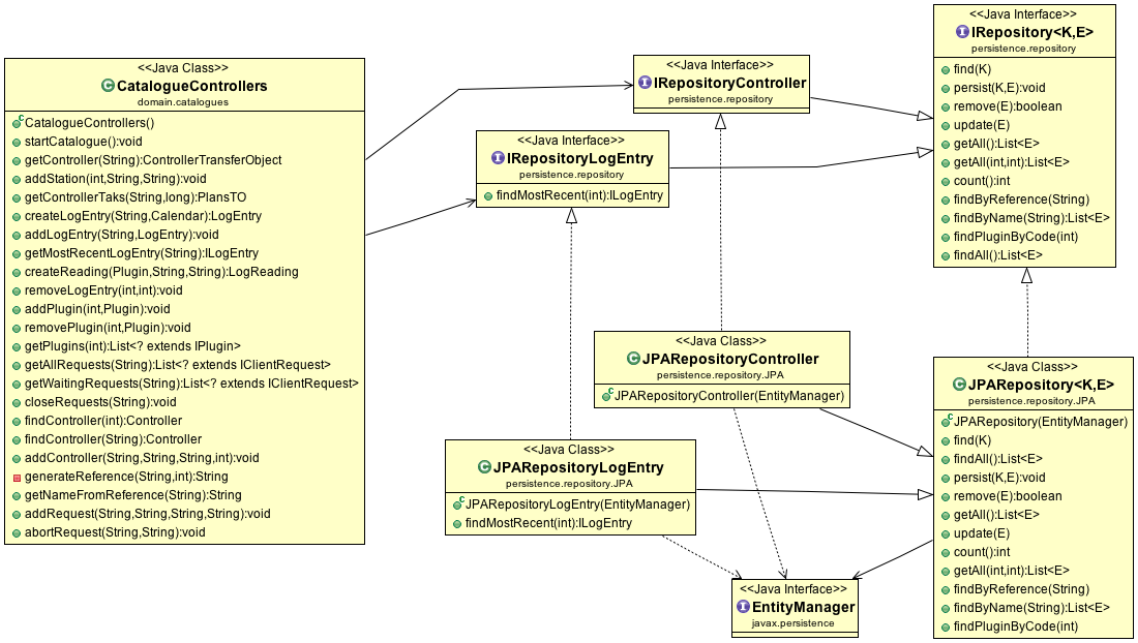


Figura 4.8: Diagrama de classes - repositórios

4.2.5 Fábricas

A utilização de interfaces e tipos genéricos, presente na organização dos repositórios, faz com que o processo de criação de um novo repositório seja mais complexo. Embora a interface "IRepository"seja comum a todos os repositórios, dependendo da entidade que se quer persistir é necessário instanciar uma classe diferente bem como o tipo da chave de pesquisa da entidade (K) e o tipo da entidade (E). Assim, para simplificar este

processo recorremos à utilização de fábricas. As fábricas são exemplos de aplicação do padrão de desenho *Creator*. Uma fábrica é responsável por criar um novo objeto que neste caso concreto será uma instância de um repositório específico. A fábrica dos repositórios encapsula a informação necessária para a criação de todos os repositórios existentes.

A possibilidade de existirem diferentes implementações da camada de persistência obriga à implementação de diferentes fábricas. Assim, a nossa aplicação tem uma interface (IRepositoryFactory) e uma implementação específica para a tecnologia JPA (JPARepositoryFactory).

Embora comparando com os repositórios, a criação das várias entidades da camada de negócio seja mais simples, seguimos o mesmo princípio tendo uma interface (IDomainFactory) e uma implementação específica (DomainFactory).

A manipulação dos repositórios e das entidades é da responsabilidade de um catálogo específico. A figura 4.9 mostra o diagrama de classes das fábricas descritas utilizadas pelo catálogo dos controladores.

Figura 4.9: Diagrama de classes - fábricas dos repositórios e das entidades

4.2.6 Inicialização dos componentes da aplicação

Já vimos que a aplicação tem diferentes repositórios para cada entidade e que podem suportar diferentes implementações da camada de persistência (abstraídas por uma interface comum). Vimos também que existem diferentes fábricas responsáveis por criar estes respositórios. Resta então saber quem é responsável por inicializar as fábricas e como é especificado o tipo de persistência em cada execução da aplicação.

A classe "AppLoader" é um EJB do tipo *singleton* que é instanciado durante o arranque da aplicação (ao contrário dos restantes EJB). O facto de ser *singleton* dita que durante o ciclo de vida da aplicação só existe uma instanciação desta classe. A fábrica das entidades é inicializada diretamente na classe "AppLoader". Para inicializar a fábrica dos repositórios é utilizada uma classe auxiliar (PersistenceFacade) para carregar informações

de um ficheiro de configuração sobre o tipo de persistência que deve ser usado no arranque da aplicação (e.g., JPA). Com base nas informações lidas, a classe "PersistenceFacade" cria uma fábrica específica que é devolvida à classe "AppLoader".

Como a classe "AppLoader" é partilhada pelos vários catálogos, estes têm também acesso às fábricas.

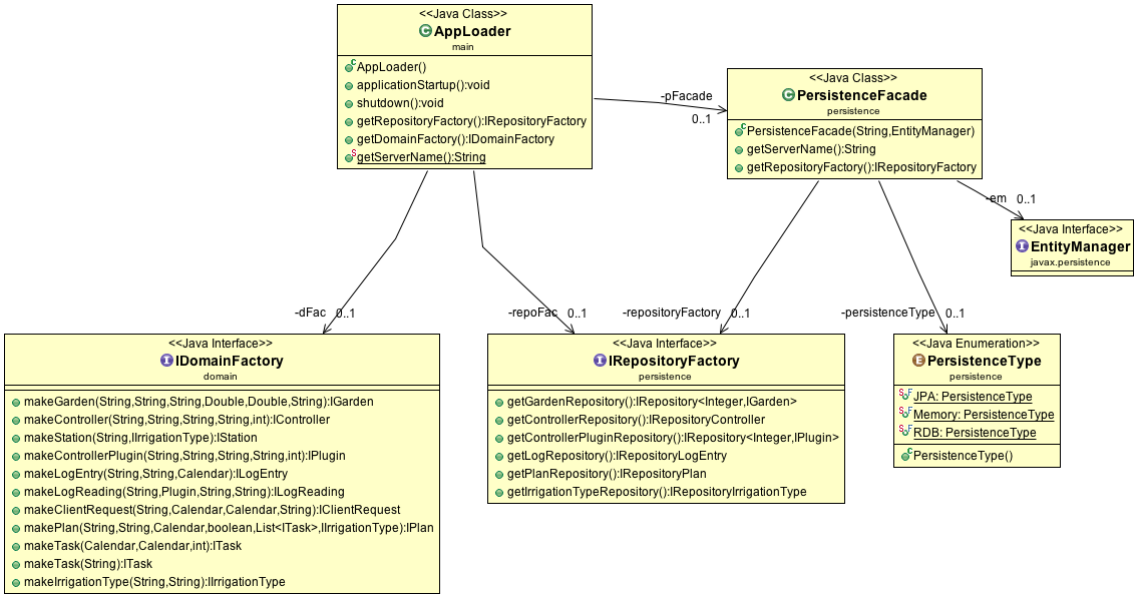


Figura 4.10: Diagrama de classes - *AppLoader* e fábricas

4.2.7 Persistência

Como já referimos na secção 4.2.4, a plataforma de gestão SRI guarda a informação numa base de dados relacional. Neste projeto utilizamos o sistema MySQL, embora outro sistema seja facilmente implementado. Para fazer a ligação a esta base de dados implementamos um repositório JPA na camada de negócio.

A Java Persistence API (JPA) oferece suporte ao mapeamento de objetos para tabelas relacionais e é constituída essencialmente por entidades de persistência (*Entities*), um gestor de entidades (*EntityManager*) e uma linguagem de interrogações (*Java Persistence Query Language (JPQL)*). A tecnologia JavaEE integra esta API na sua especificação.

Na nossa implementação, as entidades de persistência JPA são as mesmas entidades descritas na secção 4.2.1, estando mapeadas através de um ficheiro de configuração XML. Assim, as tabelas da base de dados MySQL resultam do mapeamento direto das entidades, como se pode observar no modelo Entidade-Relação em anexo (Figura B.1).

Para inicializar a aplicação com esta configuração é apenas necessário especificar no ficheiro de configuração de arranque da aplicação (ver secção anterior) a entrada "persistenceType = JPA". As informações da base de dados são injetadas automaticamente graças

à anotação "@PersistenceContext" e as operações da base de dados ficam acessíveis através da interface EntityManager já referida na secção 4.2.4.

4.2.8 Comunicação com os controladores

O sistema central escuta permanentemente pedidos de comunicação dos vários controladores. Para minimizar o tamanho da informção trocada é estabelecida uma ligação Transmission Control Protocol (TCP) tradicional através de *sockets*. Contudo, a tecnologia JavaEE possui uma arquitetura onde a programação direta de *sockets* e *threads* é abstraída do programador. Assim, para estabelecer uma ligação TCP tradicional, é preciso adicionar um conector Java EE Connector Architecture (JCA). Estes conectores são instalados no servidor aplicacional onde corre a aplicação. A aplicação consegue aceder às funcionalidades do conector embora na verdade sejam dois serviços independentes.

A solução que usamos no projeto é adaptada do projeto JCA Sockets [Jef10]. O conector JCA Sockets funciona como um servidor TCP tradicional. Sempre que um controlador se liga ao sistema central é criada uma nova ligação e o conector notifica a aplicação de um novo evento. O sistema de subscrição implementa a arquitetura JMS através da utilização de um *MessageBean* que é notificado quando ocorre uma nova ligação e recebe o objeto responsável pelo *socket* de um controlador específico. A partir deste momento a aplicação já pode comunicar diretamente com o cliente sem necessidade do conector. De seguida, descrevemos como cada pedido é processado após já estar estabelecida uma ligação.

Processamento dos pedidos

Quando existe uma ligação criada, o sistema central inicia o protocolo de comunicação enviando o primeiro pedido ao controlador. São depois trocadas diversas mensagens consoante a situação. O protocolo de comunicação está implementado com uma máquina de estados separando a conversação em fases distintas consoante o tipo de ação a ser executada.

Iniciar	O sistema central inicia a conversação enviando um pedido com o código "CGET_ALL".
Ler pedidos pendentes	O sistema central lê os pedidos guardados na base de dados que devem ser tratados.
Esperar	O sistema central espera para ser notificado de novos pedidos.
Fechar ligação	O estado dos pedidos tratados é atualizado na base de dados. A informação recebida do controlador é guardada no histórico. Finalmente, a ligação é fechada.

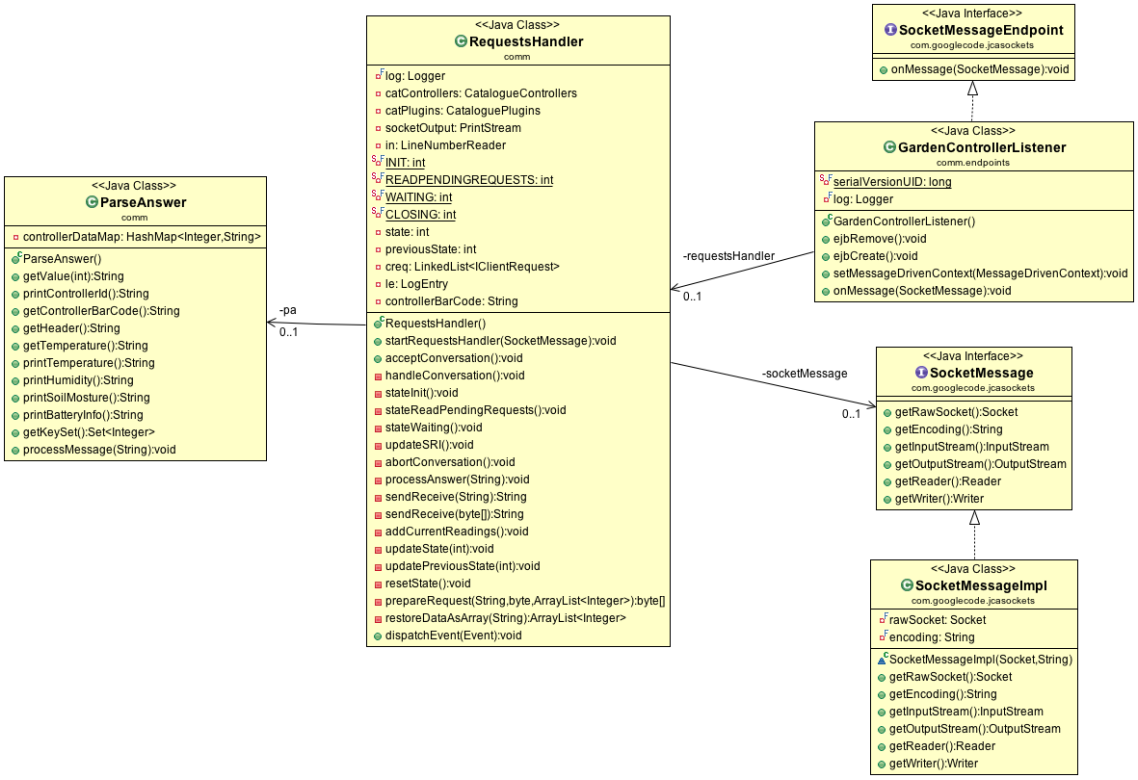


Figura 4.11: Diagrama de classes - comunicação remota com os controladores dos jardins

4.3 Camada Web

O módulo (ou camada) *Web* permite implementar um servidor *Web* dinâmico. Numa aplicação JavaEE, o módulo *Web* não tem necessariamente uma correspondência direta apenas com o módulo vista do modelo MVC. Na verdade, esta camada constitui só por si a utilização dos três módulos do MVC e utiliza o módulo EJB descrito anteriormente como auxiliar do módulo controlador. Nesta implementação, as *Servlets* são utilizadas como controladores e as páginas Java Server Pagess (JSPs) são responsáveis pela apresentação.

4.3.1 Tratamento de um pedido *Web*

As *Servlets* estão organizadas em duas camadas: controladores *Web* (Handlers) e eventos *Web* (Event).

Os controladores *Web* seguem o mesmo princípio dos controladores descritos na secção 4.2.2 e são a porta de entrada dos pedidos *Web*. Sempre que é feito um pedido de página *Web* ao servidor, existe um controlador *Web* específico para esse URL. Cada controlador *Web* faz um novo pedido interno que é apanhado por um evento *Web*, responsável pelo tratamento desse pedido.

Um evento *Web* analisa o pedido e se necessário faz a chamada à camada de negócio invocando um ou mais dos controladores do domíno disponíveis. Finalmente, este evento

Web invoca a página JSP que é responsável por constuir a página HTML passando, se for o caso, uma referência para o controlador da camada de negócio.

O esquema 4.12 ilustra esta hierarquia:

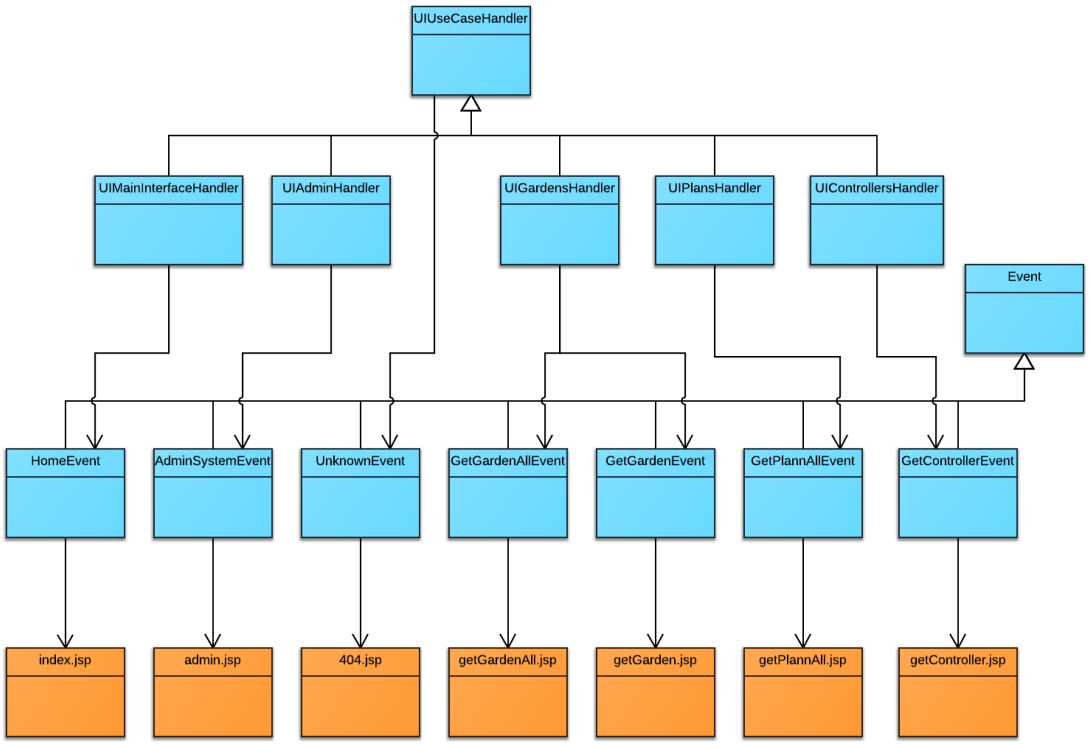


Figura 4.12: Diagrama de Classes e JSP - Pedidos Web

Acesso à camada de negócio

Como referido na secção 4.2.2, o módulo EJB contém controladores preparados para receber eventos de outros módulos quer estejam na mesma máquina ou remotamente. Isto é possível, pois cada controlador tem uma interface específica. Na aplicação desenvolvida, o módulo *Web* é instalado em conjunto com o módulo EJB na mesma máquina e por isso as interfaces dos controladores contêm a anotação "@Local". A utilização destas interfaces permite que os métodos dos controladores possam ser usados nas *Servlets* e nas páginas JSP.

Objetos de transferência

Para além dos controladores, é necessário, na maioria das vezes, utilizar os dados de uma ou mais entidades. Os objetos de transferência (*Transfer Objects*) permitem transferir a totalidade ou parte da informação guardada por uma entidade do domínio para a camada *Web* ou diretamente através de um serviço na *Web*. O facto desta transferência poder

ser remota obriga à serialização deste tipo de objetos. Estes objetos são construídos na camada de negócio, mas são importados para o módulo *Web* como bibliotecas.

Exemplo - Interface para mostrar todos os planos de rega

A figura 4.13 ilustra a sequência de chamadas desde o momento que é feito o pedido *Web*. Neste esquema omitimos a parte do domínio e o retorno da informação focando apenas no sentido das chamadas da aplicação *Web* à camada de negócio.

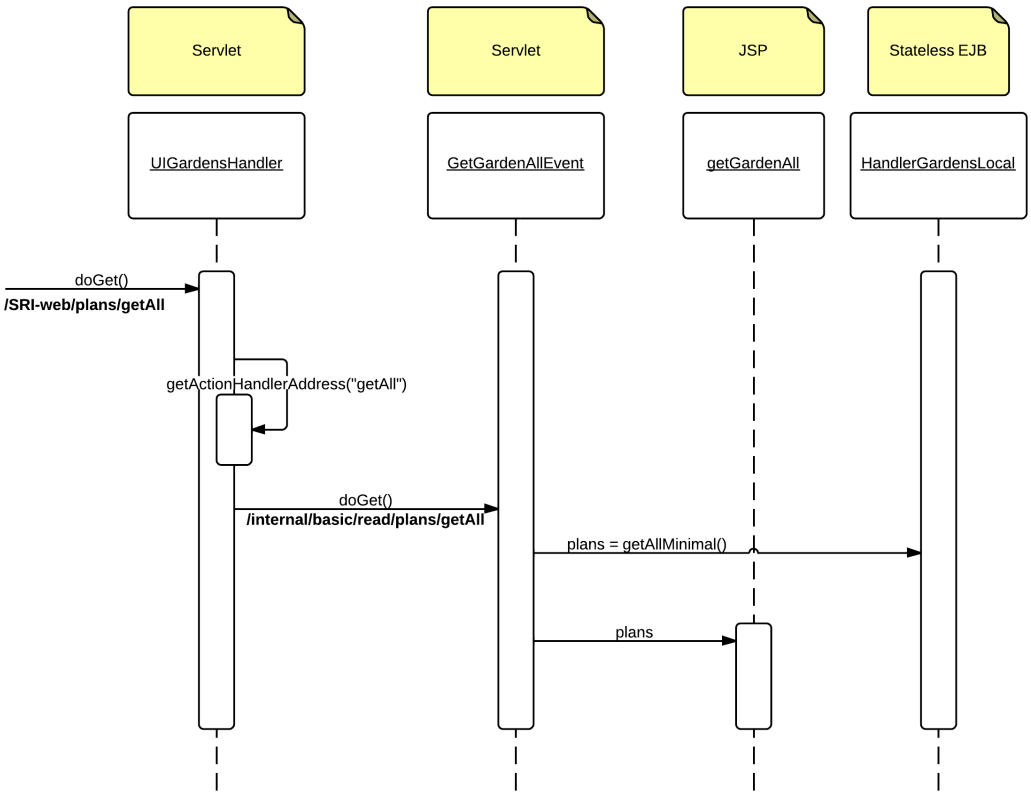


Figura 4.13: Diagrama de Sequência - Obter todos os planos de rega

Capítulo 5

Plataforma de Gestão

O sistema de rega SRI é composto por um conjunto de controladores e por uma plataforma central de gestão, sendo responsável por gerar os conteúdos da interface *Web*, efetuar as operações do domínio, guardar a informação numa base de dados e assegurar a comunicação com os controladores. Para além da implementação da plataforma são necessários alguns passos para a sua configuração e instalação, sendo a segurança informática uma das preocupações principais. Este capítulo descreve a arquitetura do sistema, tecnologia utilizada na preparação da aplicação para produção e ainda a ilustração das principais funcionalidades da plataforma de gestão.

5.1 Arquitetura

A Figura 5.1 ilustra a arquitetura que propômos. O nosso sistema é constituído por uma plataforma central de gestão *online* (A) a partir da qual é possível comunicar com vários controladores (C1, C2, ..., Cn) através de comunicação móvel (B) (e.g., GPRS). Em cada jardim haverá tipicamente um controlador que será responsável por vários tipos de sensores (D1, D2, ..., Dn) e várias válvulas solenóides (eletroválvulas) (E1, E2, ..., En).

A plataforma está disponível para o operador a partir de um navegador *Web*, mas versões posteriores poderão incluir outro tipo de clientes como, por exemplo, *Desktop* ou dispositivos móveis (e.g., *Android*). As principais funcionalidades são: agendamento de rega em cada jardim; efetuar operações remotas manuais, como ligar ou desligar imediatamente o sistema de rega; e administrar o sistema. Os controladores são programáveis exclusivamente através da plataforma e os principais requisitos são: comunicação remota, baixo consumo, execução de um plano de rega e leitura de sensores.

5.2 Instalação

Uma aplicação JavaEE corre num servidor aplicacional. Neste projeto utilizamos o servidor Glassfish [Ora] e a nossa aplicação é distribuída num único ficheiro, um *Enterprise*

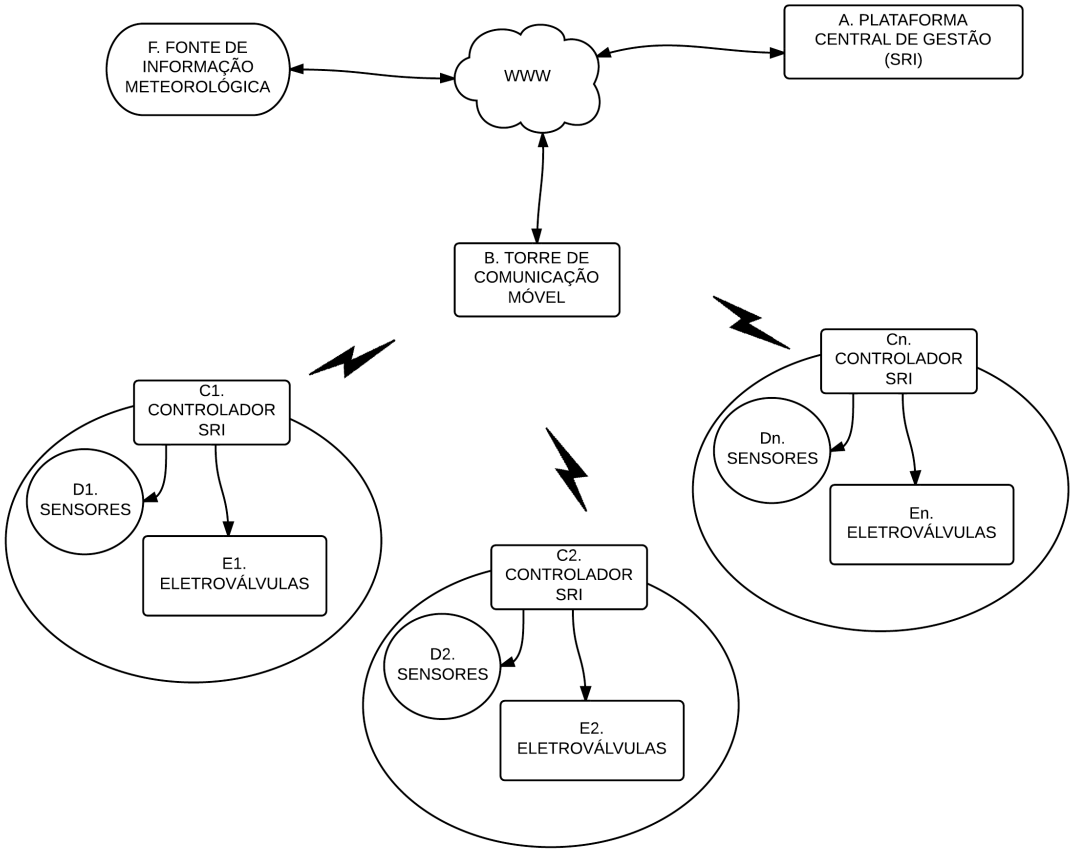


Figura 5.1: Arquitetura geral do sistema

ARchive (EAR), que contém um módulo *Web* e um módulo EJB.

Para instalarmos e correrrmos a aplicação de gestão utilizámos o serviço de computação na *nuvem Amazon EC2* da Amazon [Ama]. Este serviço disponibiliza servidores virtuais remotos, disponíveis vinte e quatro horas por dia e acessíveis com IP público. Esta máquina já vem configurada com um sistema operativo, no nosso caso, utilizámos um sistema Linux, Ubuntu Server [Ubu]. Após instalarmos e configurarmos o servidor aplicacional *Glassfish*, fazemos *deploy* do ficheiro EAR, que contém a aplicação e informação de instalação destinada ao *Glassfish*, e a aplicação fica pronta a utilizar.

5.3 Ilustração das funcionalidades

Nesta secção mostramos o aspeto da aplicação relativamente a algumas funcionalidades, bem como a interação com o utilizador.

5.3.1 Jardins

Mostrar os jardins

Os jardins são espaços físicos georeferenciáveis. Optámos por utilizar a *GoogleMapsAPI* [Goo13b] para mostrar a localização de cada jardim num mapa. A figura 5.2 mostra as marcas dos jardins em cima do mapa nas localizações respetivas. Após o carregamento do mapa, as marcas são automaticamente preenchidas a partir da base de dados utilizando um serviço na *Web*. Quando este carregamento é feito, apenas a informação básica de cada jardim é devolvida.

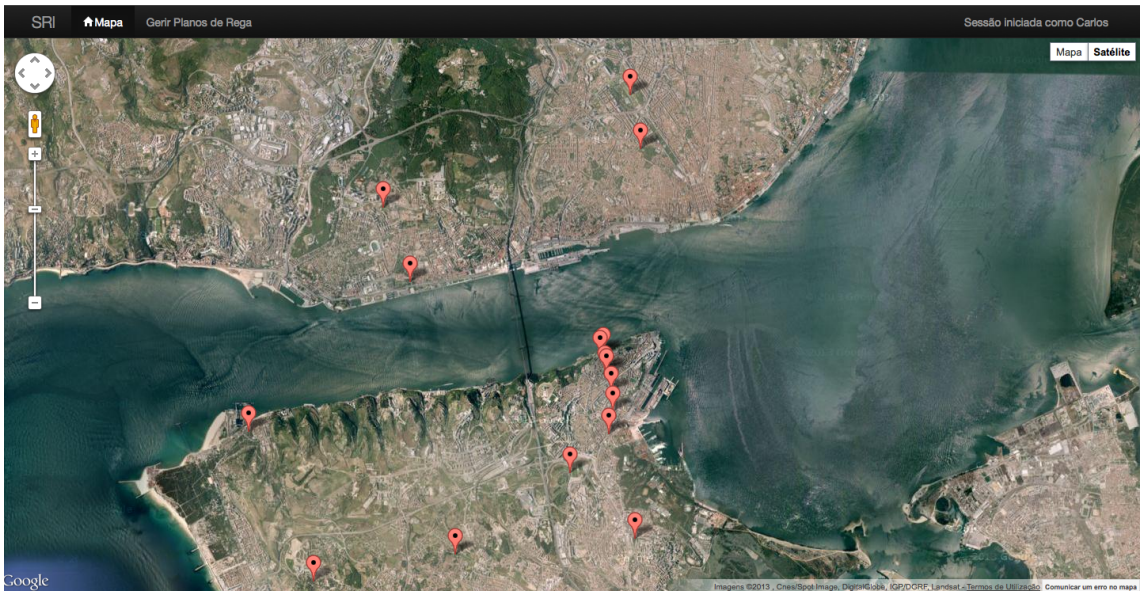


Figura 5.2: Interface com mapas da *Google*, marcas dos jardins e menu na barra superior

Seleccionar e gerir um jardim específico

Para obter mais informações de um jardim pode selecionar-se a marca do mapa que lhe corresponde. Nessa altura, o sistema carrega os dados específicos daquele jardim (Figura 5.3). Associada à informação do jardim estão duas opções: "Gerir Jardim" e "Remover Jardim".

Se escolhermos a opção "Gerir Jardim" é apresentada uma janela modal por cima do mapa com informação detalhada do jardim, incluindo a sua lista dos controladores (Figura 5.4). A lista de controladores mostra informações básicas de cada controlador, se há algum alerta e dá acesso à interface de gestão do controlador.

5.3.2 Planos de rega

Numa situção real, a plataforma de gestão SRI pode assegurar a rega de centenas de jardins. Cada jardim pode ter mais do que um controlador e estes, por sua vez, podem ter

Figura 5.3: Obter informação de um jardim específico

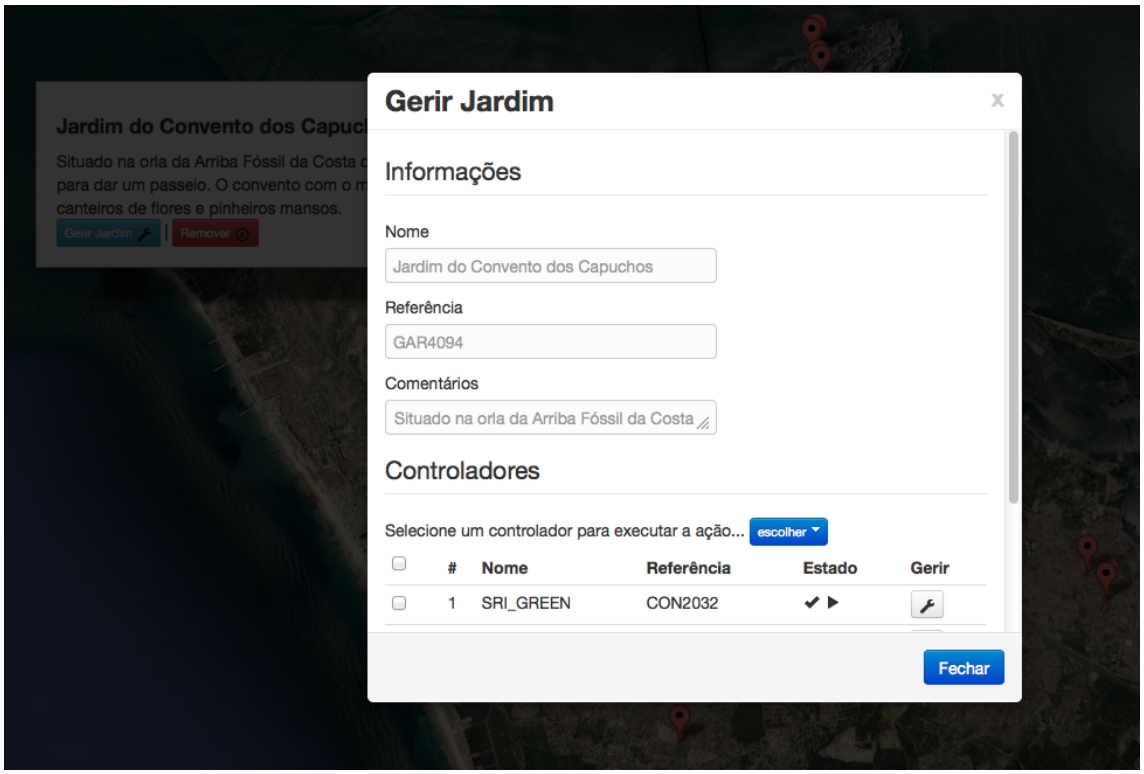


Figura 5.4: Janela de gestão de um jardim

várias estações de rega. Uma das responsabilidades do operador do sistema é programar os controladores para que cada estação abasteça a sua área de rega com a quantidade de água necessária. Facilmente se percebe que esta tarefa pode tornar-se extremamente repetitiva, morosa e conducente a erros. O que acontece tipicamente é que o processo de rega é o mesmo para várias estações de controladores instalados em jardins com condições semelhantes.

A plataforma SRI pretende automatizar ao máximo a gestão da rega nos vários jardins facilitando a programação dos controladores em larga escala. O sistema que propômos prevê a atribuição de um plano de rega a um tipo de rega, ao invés de ser específico por estação. Cada estação tem obrigatoriamente um tipo de rega e por isso, ao associarmos um plano de rega a um tipo de rega estamos automaticamente a programar todas as estações desse tipo de rega. Esta organização simplifica ainda o número de tarefas que são guardadas no sistema. Por exemplo, independentemente do número de estações de rega, se existirem três tipos de rega cujos planos contenham cada um quatro tarefas, são apenas necessárias guardar doze tarefas no total.

5.3.3 Gestão dos planos de rega

A partir da barra de menu superior podemos aceder à gestão dos planos de rega. Esta interface permite escolher um plano já existente e configurar as suas tarefas num calendário. A programação de um plano de rega é sempre referente a um ciclo de uma semana e por isso não interessa identificar qual a semana do ano, mas sim quais as tarefas de cada dia da semana.

A interface de calendário permite criar novas tarefas selecionando a hora de início e arrastando o bloco até à hora de fim. Mais tarde podemos alterar o tamanho do bloco ou arrastá-lo para outro horário. Quando terminarmos as alterações, selecciona-se a opção "Ações" seguida de "Guardar". Se, pelo contrário, quisermos reverter as alterações, selecionamos a opção "Repôr".

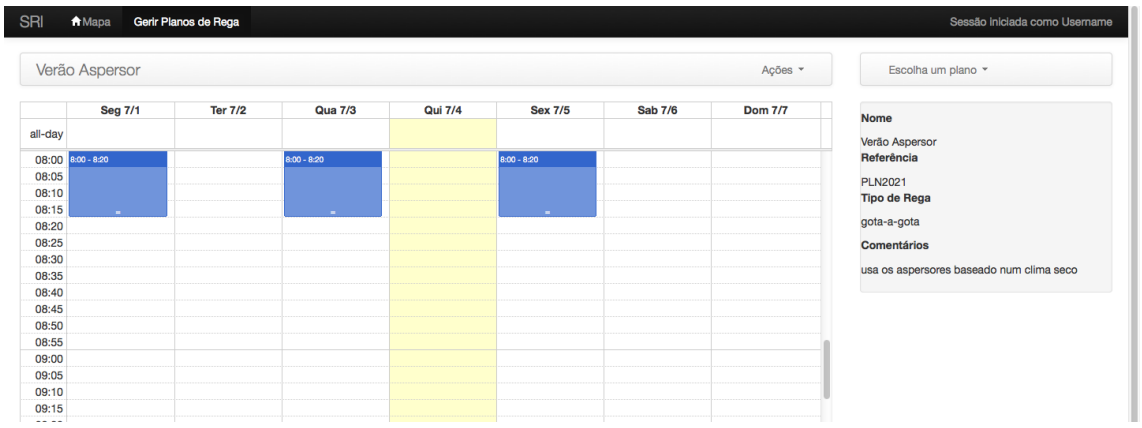


Figura 5.5: Gestão de um plano de rega seleccionado

5.3.4 Criar e remover um plano de rega

Para criar um novo plano, selecionamos a opção "Planos" seguida de "Novo plano". Uma janela modal é então apresentada (figura 5.6) requerendo as informações a preencher. Na criação de um novo plano é ainda obrigatório definir qual o tipo de rega a que este deve

ser atribuído e qual o dia do ano em que deve entrar em vigor. Um plano não tem data de fim, evitando assim que, em caso de má configuração, as estações daquele tipo fiquem sem qualquer tipo de rega associado. Pelo mesmo motivo, um plano só pode ser removido se não estiver em vigor.

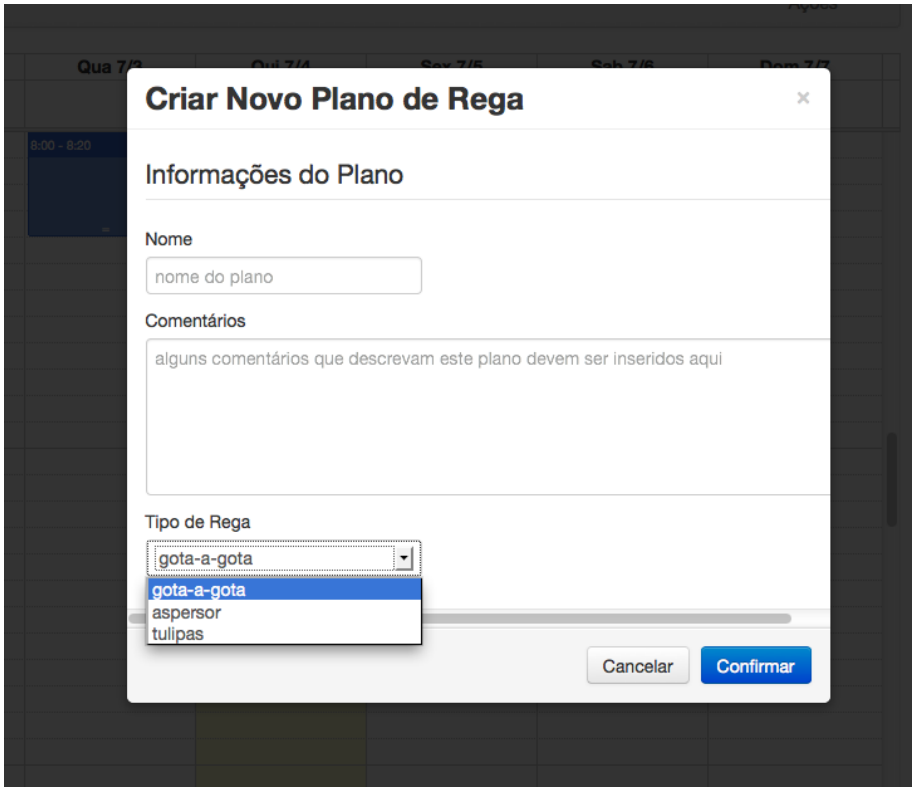


Figura 5.6: Criar um novo plano

5.3.5 Plano de rega em vigor

A plataforma SRI permite que cada tipo de rega tenha vários planos associados, mas apenas um se encontra ativo para um determinado período de tempo. Esta organização permite ter um histórico de planos atribuídos a um tipo de rega, o que pode facilitar a programação da rega se existir uma mudança cíclica.

A associação do plano a um tipo de rega permite definir a data de entrada em vigor. Quando é atingida essa data, o plano previamente agendado fica automaticamente em execução sobrepondo-se ao plano previamente em vigor. Em qualquer altura é possível ativar imediatamente um plano já existente, o que na prática significa alterar a sua data de entrada em vigor.

Finalmente, sempre que um plano entra em vigor ou o plano em vigor é alterado, é criado um novo pedido para enviar aos controladores respetivos (ver secção 4.2.8). Este pedido é enviado imediatamente se o controlador estiver *online* ou agendado para enviar mais tarde.

5.3.6 Gestão do Controlador

Quando selecionamos a opção "Gerir Controlador" (ver secção 5.3.1) é apresentada uma interface para gerir o controlador num novo separador do *browser*. A razão para abrir um novo separador justifica-se se for necessário gerir simultaneamente vários controladores.

A interface de gestão do controlador é constituída por a) visão semanal do plano de rega (eventos), b) informações detalhadas do controlador, c) lista de estações e d) última leitura dos sensores. A Figura 5.7 ilustra esta interface.

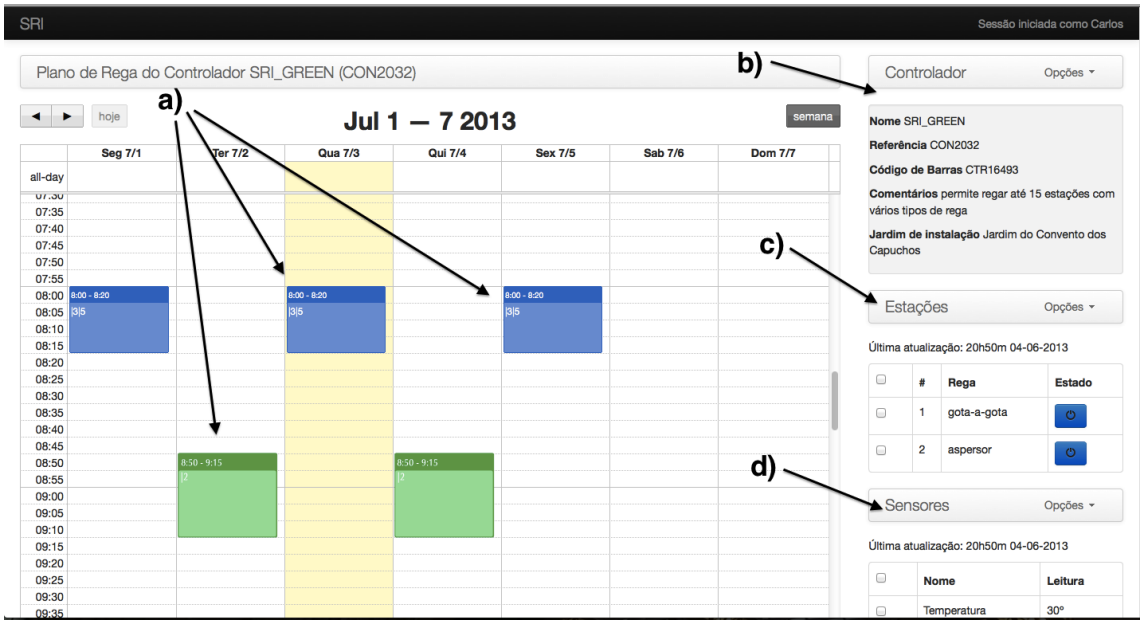


Figura 5.7: Gestão do controlador

Estações e Calendário

Quando é atribuída uma nova estação ao controlador é obrigatório definir o seu tipo de rega. Isto permite atribuir automaticamente o plano de rega que está ativo nesse momento para aquele tipo de rega (ver secção 5.3.2). Assim, ao controlador estão associados indiretamente vários tipos de rega e o calendário de cada controlador apresenta a combinação das tarefas dos planos de rega respetivos. Por esta razão, a visão do calendário na interface de gestão do controlador é apenas de leitura.

A visão semanal tem uma data concreta, ao contrário da interface de gestão dos planos (ver secção 5.3.5). Isso permite ao operador consultar os planos atualmente em vigor, bem como o histórico de planos.

Cada bloco de rega tem os números das estações associadas, a cor do tipo de rega e o nome do plano, facilitando a distinção dos vários planos. Contudo, um bloco desenhado para uma tarefa de rega pode, na prática, representar vários eventos de rega para o controlador SRI. Isto acontece se existirem várias estações com o mesmo tipo de rega, pois

estas nunca estão ativas em simultâneo. Neste caso há uma sobreposição de eventos, a plataforma é responsável por ajustar automaticamente a hora de início do evento.

Por exemplo na figura 5.7 o bloco azul da segunda-feira determina uma ação de rega de 20 minutos que inicia às 8h. No entanto, como este bloco pertence a duas estações, o plano de rega que é enviado ao controlador terá um evento das 8h-8h20 para a estação 3 e das 8h20-8h40 para a estação 5.

Se quisermos desativar uma estação por tempo indeterminado podemos dizer que ela é de um tipo especial que nunca tem nenhum plano de rega atribuído.

Para além da rega automática, as estações podem ser controladas manualmente quando o controlador está *online*. As ações que podem ser desencadeadas são:

- Parar agora: esta opção permite parar imediatamente a rega numa estação. Esta acção tem efeito somente no evento do plano que está em execução nesse momento;
- Iniciar agora: esta opção permite iniciar imediatamente a rega numa estação. Para evitar que a rega se prolongue indefinidamente por descuido, existe um tempo máximo de duração estabelecido.
- Temporizador: agendar um evento especial que se sobrepõe ao plano de rega. Este evento tem uma data de início e uma data de fim e permite que uma estação seja desativada durante um período de tempo limitado. Após esse período de tempo é retomado o plano de rega em vigor.

As acções de iniciar ou parar imediatamente a rega são úteis para testar o bom funcionamento das válvulas solenóides e para efeitos de demonstração. A funcionalidade de temporizador é útil se uma zona do jardim estiver temporariamente em obras, por exemplo.

Sensores

Na interface de gestão é ainda possível consultar a última leitura dos sensores. Se algum dos sensores apresentar valores fora do esperado, o operador será notificado através de uma mensagem. A secção 8.1 explora a possibilidade futura de aceder a todas as leituras registadas ao longo do tempo.

Como cada controlador pode ter uma configuração diferente de *hardware*, a interface permite adicionar e remover sensores como se fossem *plugins*.

As opções das estações e dos sensores podem ser acedidas selecionando o botão "Opções" como mostram as figuras 5.8 e 5.9.

Figura 5.8: Opções para as estações

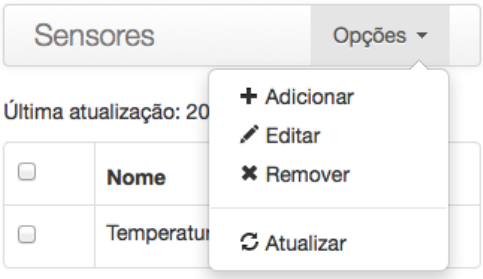


Figura 5.9: Opções para os sensores

Capítulo 6

Protótipo

Neste capítulo descrevemos detalhadamente a criação de um protótipo para o controlador SRI. Este protótipo é composto por uma peça de *hardware* que prentende estudar a viabilidade do projeto. É criado antes do produto final, sendo por isso uma peça temporária e mais flexível nas suas características.

O desenvolvimento de um protótipo que implementa as funções básicas de um controlador de rega tornou-se especialmente importante, contribuindo para ganhar a confiança do cliente e para validar a solução que propomos. De seguida relembramos as funcionalidades principais requeridas para o controlador:

- comunicação com o sistema central através de GPRS;
- execução de um plano de rega através da gestão de várias válvulas solenóides;
- interação com diversos sensores;
- baixo consumo energético;
- ativação manual (ação através de botão local ao controlador);
- dimensões reduzidas;
- preço económico.

Nas próximas secções detalhamos os vários componentes escolhidos para a construção do protótipo, o desenho do circuito, a programação do microcontrolador e finalizamos apresentando alguns resultados e otimizações.

6.1 Controlador Arduino Uno

A criação de um protótipo com os requisitos acima referidos obriga, em primeiro lugar, à utilização de um microcontrolador [Tan06] capaz de interagir com vários componentes de entrada e saída (E/S), como é o caso dos sensores ou das válvulas solenóides. Tipicamente

um microcontrolador encontra-se embutido numa Placa de Circuito Impresso (PCI) onde estão inseridos também outros componentes. Usualmente esta placa possui um conjunto de portas onde é possível ligar dispositivos externos.

Atualmente, com a expansão do movimento *Internet of Things* [Pfi11], existe no mercado uma oferta variada de controladores de uso geral que já contêm um microcontrolador e facilidades de ligação a outros componentes de *hardware*. Para a criação do protótipo optámos pela utilização do controlador Arduino Uno, que pertence à família Arduino [Ard13]. A escolha da plataforma Arduino prendeu-se sobretudo com a sua larga utilização, com as várias bibliotecas de *software* já implementadas, a excelente documentação, os preços acessíveis e, finalmente, por ser uma plataforma de desenvolvimento aberta.

Em baixo descrevemos brevemente as especificações do Arduino Uno, que serão úteis para perceber as várias otimizações que foram feitas numa fase mais avançada do projeto, abordadas no capítulo 7.

6.1.1 Microcontrolador

Um microcontrolador [Tan06] é um circuito integrado com capacidade de processamento, memória e entrada e saída. O Arduino Uno possui um *chip* ATMEL AVR ATMEGA328P [ATM] como microcontrolador principal, do qual se destacam as principais características:

- utiliza a arquitetura *Harvard* modificada [Sin08];
- três memórias distintas: 1K EEPROM (não volátil, memória de dados auxiliar), 32K FLASH (não volátil, onde é guardado o *bootloader* e o código a ser executado) e 2K SRAM (volátil, memória de execução principal);
- velocidade de relógio entre 1MHz e 20MHz;
- instruções RISC;
- oscilador interno e possibilidade de oscilador externo;
- vários modos de *sleep*;
- temporizador programável *Watchdog Timer*;
- interrupções externas com controlo do tipo de sinal (*Interrupt Sense Control*) em dois pinos;
- auto-reset;
- tensão de entrada entre 1,8V e 5,5V;
- baixo consumo: 240μA @ 1,8V (ligado) ou 0,1μA @ 1,8V (adormecido) com velocidade de relógio de 1MHz;
- reprogramação (*In-System Programming (ISP)*) [Mic] utilizando o protocolo *Serial Peripheral Interface (SPI)* [Ard].

No caso específico do Arduino Uno, o microcontrolador contém as seguintes configurações:

- 28 pinos (pacote Dual Inline Package (DIP), modelo ATMEGA328P-PU);
- operação a 5V;
- velocidade de relógio de 16MHz com oscilador externo;
- *bootloader* ocupa 0,5KB da memória FLASH.

A figura 6.1 ilustra o microcontrolador ATMEGA328P-PU.

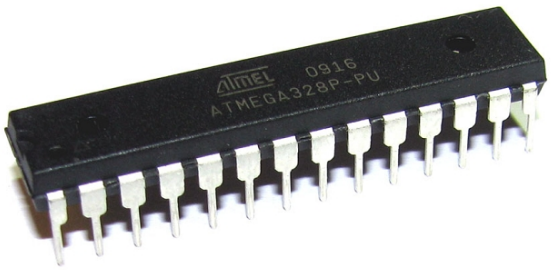


Figura 6.1: Microcontrolador ATMEGA328P-PU

6.1.2 Entrada e Saída

O controlador Arduino Uno possui um conjunto de portas que abstraem e simplificam a interação (entradas e saídas) com o microcontrolador. Cada uma destas portas está mapeada para um ou mais pinos do microcontrolador. Em baixo descrevemos resumidamente as portas mais importantes, ilustradas na Figura 6.2.

Portas digitais As portas digitais permitem ligar dispositivos de entrada ou saída e ler ou escrever valores iguais a 0 ou a 1 (tensão respetiva de 0V (*ground*) e 3.3V/5V). São especialmente úteis para controlar ações em dispositivos como, por exemplo, ativar a ligação de uma válvula solenóide.

Portas especiais Algumas das portas presentes no controlador têm funções extra específicas, como por exemplo:

- possibilidade de interrupção externa ao microcontrolador (D2 e D3);

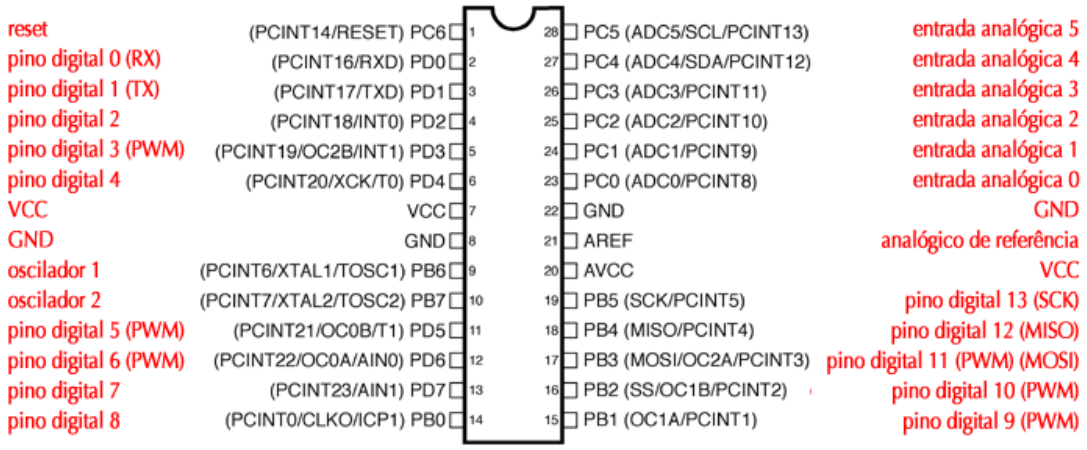


Figura 6.2: Mapeamento dos pinos do micontrontrolador para as portas do Arduino Uno

- possibilidade de simular uma saída analógica numa porta digital através da utilização de *Pulse With Modulation (PWM)* (D3, D5, D6, D9, D10, D11);
- utilização de protocolos de comunicação (ver secção 6.1.2);
- *reset* externo ao microcontrolador (RST).

Portas analógicas As portas analógicas, só de entrada, (A0 - A5), permitem aplicar uma tensão de entrada nos pinos do microcontrolador (entre o *ground* e ~5V) e mapear esse valor para um número inteiro entre 0 e 1023. São especialmente úteis para ler valores de sensores como humidade do solo ou luminosidade, cujos valores são fornecidos de forma analógica.

Portas de alimentação O controlador escolhido tem uma tensão nominal de operação de 5V e permite uma alimentação no intervalo entre os 5V e os 12V de acordo com a sua especificação [Ard12]. A utilização de valores de tensão superiores a 5V é possível pois o controlador possui um conversor de tensão incorporado. A alimentação pode ser feita de quatro formas:

- aplicando tensão na porta "5V" sem utilização do conversor de tensão;
- aplicando tensão na porta "VIN" (ou "RAW") (até 12V), que utiliza um conversor de tensão interno;
- utilizando um *jack* de 2,1mm (até 12V), que utiliza o conversor de tensão;
- através da ligação USB (cerca de 5V).

UART ou *USART*	O protocolo *UART* permite a comunicação em série, assíncrona e requer duas portas (RX - Recetor e TX - Transmissor, portas D0 e D1, respetivamente) e uma massa comum (GRD ou *ground*) na sua configuração mais básica. Este protocolo obriga a que os dois comunicadores utilizem a mesma taxa de transmissão e tem limitações na velocidade de transmissão máxima.
I2C (TWI)	O protocolo *Inter-Integrated Circuit (I2C)* (portas A4 e A5) permite a transmissão de dados síncrona e utiliza duas linhas para a comunicação entre dois ou mais dispositivos.
SPI	O protocolo *SPI* estabelece uma comunicação em série, síncrona e permite a utilização de vários dispositivos periféricos ao mesmo tempo. Existe a noção de *master* e *slave* em que há três linhas (MISO, MOSI e SCK), que são partilhadas por todos os dispositivos, e uma linha específica (SS) para cada periférico (*slave*). No controlador Arduino Uno, este protocolo pode ser utilizado através das portas 10 (SS), 11 (MOSI), 12 (MISO) e 13 (SCK) ou através de conectores dedicados disponíveis no controlador. A instalação de um *bootloader* no microcontrolador pode ser feita utilizando este protocolo.
USB	O protocolo *Universal Serial Bus (USB)* pode ser utilizado graças à adição de outro microcontrolador (ATmega16U2), que faz a conversão de USB para Série (mapeando a comunicação para os pinos RX e TX do microcontrolador ATMEGA328P). O microcontrolador adicional é programado especificamente para fazer a comunicação USB e possui *firmware* que permite fazer a ligação a um computador tal como um dispositivo USB *plug and play* comum.

Tabela 6.1: Protocolos de comunicação suportados pelo Arduino Uno

As portas de alimentação também podem ser usadas para alimentar outros dispositivos. Por exemplo, se se fornecer uma tensão de 9V na porta "VIN", a porta "5V" fornecerá uma tensão de saída de 5V e a porta "3V" uma tensão de saída de 3V. Outras combinações também são possíveis.

Cada porta de Entrada/Saída apresenta uma intensidade de 50mA.

Portas para comunicação O modelo Arduino Uno contém um conjunto de portas especiais que possibilitam a utilização de vários protocolos para comunicação com um computador, outro Arduino ou outros microcontroladores e dispositivos periféricos em geral. Estes protocolos são implementados pelo microcontrolador referido na secção 6.1.1. A tabela 6.1 descreve brevemente estes protocolos.

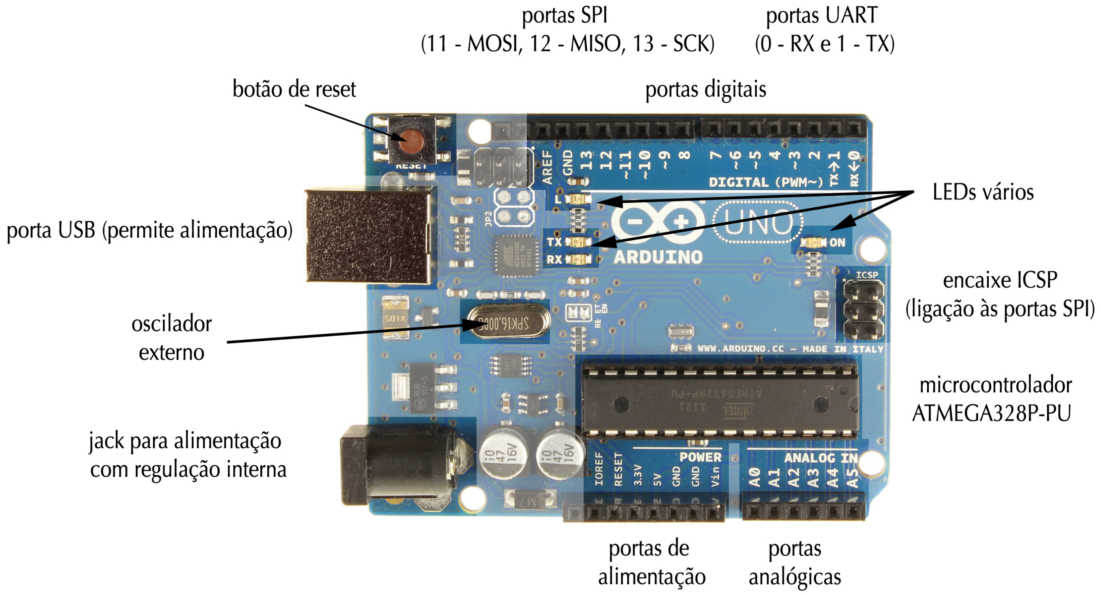


Figura 6.3: Portas e componentes principais do Arduino Uno

6.2 *Hardware* complementar

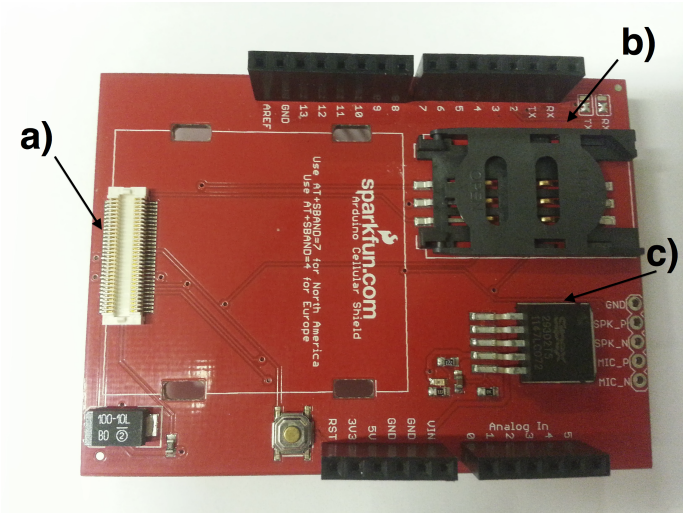
Dada a natureza de utilização geral do controlador Arduino Uno, este não tem a capacidade de comunicar via GPRS ou controlar o fluxo de água, nem possui sensores específicos úteis para monitorizar diversos parâmetros relacionados com a rega. É por isso necessário juntar *hardware* extra com funções específicas, mas capaz de interagir com o controlador através das várias portas (ver secção 6.1.2). Mesmo no caso de um controlador mais específico, há vantagens em ter alguns componentes externos substituíveis para permitir uma maior compatibilidade e facilidade de manutenção. Além disso, existem restrições físicas, por exemplo no caso das válvulas solenóides, que estão, na maior parte das vezes, a uma distância considerável do controlador. Descrevemos de seguida os componentes adicionados.

6.2.1 Módulo GSM/GPRS

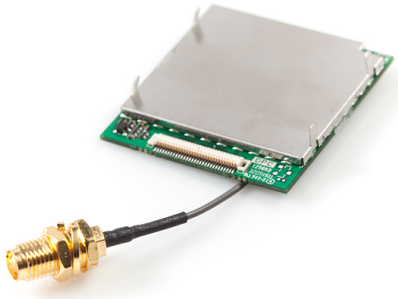
Para comunicar via GPRS é necessário um módulo que suporta a comunicação GSM e GPRS. O módulo SM5100B é um exemplo destes dispositivos, mas possui conetores desenhados para uma instalação Surface Mount Device (SMD) não sendo indicado para prototipagem manual. Este cenário é bastante comum e existem por isso placas (*shields*) no mercado que permitem adaptar o módulo ao controlador Arduino sem serem necessárias ligações extra com fios. É o caso da placa SM5100B da *SparkFun Electronics* [Spa12] que usamos neste protótipo. Como se pode ver na figura 6.4a, esta placa incorpora *a)* o

conetor do módulo, *b)* um conetor para introdução do cartão Subscriber Identity Module (SIM) e *c)* um conversor de tensão. O módulo (figura 6.4b) já possui um conetor que permite a ligação de uma antena (figura 6.4c).

Apesar das dimensões muito reduzidas do conetor do módulo GSM/GPRS, a placa apresenta a mesma disposição de portas que é encontrada no Arduino Uno, mapeando as ligações entre os pinos do microcontrolador do Arduino Uno e o módulo GSM/GPRS (figura 6.4).

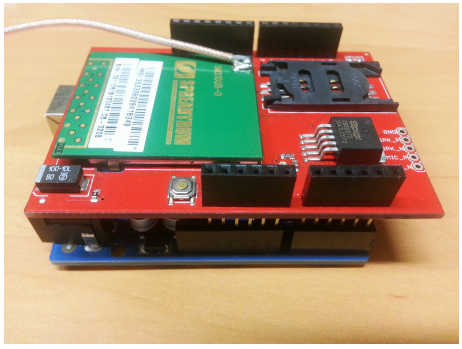


(a) Placa sem módulo



(b) Módulo GSM/GPRS

(c) Antena



(d) Placa acoplada ao Arduino Uno

Figura 6.4: Pormenores da placa SM5100B

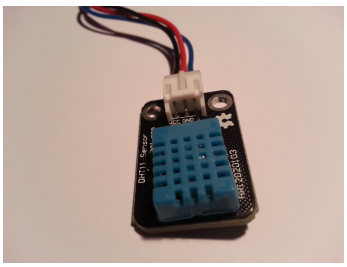
6.2.2 Sensores

Para monitorizar diversos parâmetros relacionados com a rega, são utilizados sensores que podem ser ligados ao controlador. A tabela 6.2 descreve os sensores utilizados neste projeto, embora no futuro possam ser adicionados outros tipos de sensores. Os sensores de temperatura, humidade e luminosidade recolhem informação sobre as condições ambientais de cada jardim, que pode ser utilizada para controlar a execução do plano de rega. O objetivo do caudalímetro é detetar fugas de água, problema que afeta diversas instalações

Nome	**Características**	**Alimentação**	**Tipo de Leitura**
Temperatura e Humidade do ar	medição da temperatura (-10 a 50 graus celsius) e da humidade relativa (20% a 90%)	3,3V–5V	digital
Humidade do solo	medição entre 0 e 950: solo seco $[0, 300]$; solo húmido $[300, 700]$; água em excesso > 700	3,3V–5V	analógica
Luminosidade	a resistência do sensor varia em função da quantidade de luz	3,3V	analógica
Caudalímetro	mede o caudal de água que passa num tubo (mede o fluxo entre 1 e 30 litros por minuto).	5V–18V	digital

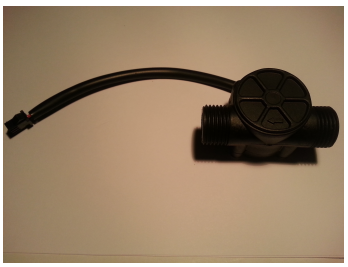
Tabela 6.2: Descrição dos sensores utilizados no controlador SRI

de rega. A figura 6.5 ilustra alguns dos sensores utilizados e a secção 6.3 descreve a utilização do sensor de temperatura e humidade do ar em conjunto com o controlador.



(a) Temperatura e Humidade

(b) Humidade do solo



(c) Caudalímetro

Figura 6.5: Ilustração dos sensores

Válvula solenóide

Outro componente complementar ao controlador é a válvula solenóide. A válvula é essencial pois permite ativar e desativar a passagem de água numa determinada zona do jardim. De acordo com o cliente, o controlador SRI deve ser desenhado de forma a ser compatível com as válvulas solenóides já existentes no mercado. Dessa forma evitar-se-iam alterações invasivas aos sistemas de rega já implantados nos jardins públicos.

A válvula escolhida para utilização com o protótipo é da marca *RainBird* [Rai13a] (figura 6.6) e é utilizada atualmente em muitas instalações de rega de jardins públicos em Portugal. A secção D em apêndice descreve o funcionamento eletromecânico deste tipo de válvulas e qual o circuito necessário para a sua utilização neste projeto.

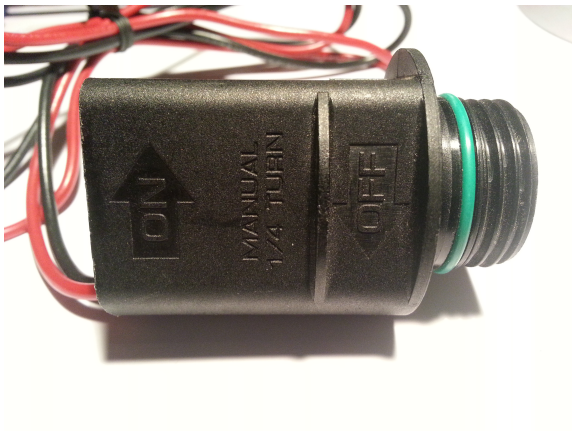


Figura 6.6: Válvula solenóide da empresa *RainBird*

6.3 Desenho do circuito

Para o desenho do circuito do protótipo utilizámos uma ferramenta de *software* chamada Fritzing [Fri13]. A figura 6.7 mostra o desenho do circuito com o controlador Arduino Uno e a placa GSM/GPRS. Mostra também os pontos de ligação dos sensores de temperatura, humidade do ar e luminosidade, fontes de alimentação e válvula solenóide. Além disso são visíveis componentes auxiliares (resistências, díodo e transístor) para efetuar estas ligações. Na solução final serão utilizadas seis válvulas solenóides por controlador, mas para efeitos de protótipo considerou-se apenas uma.

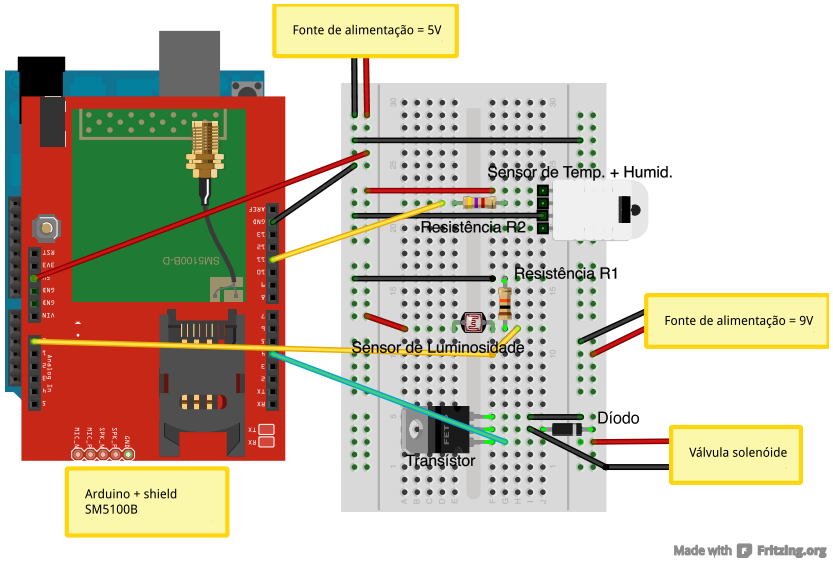


Figura 6.7: Circuito completo do protótipo

6.3.1 Alimentação

De acordo com os requisitos do cliente, o produto final deverá ter autonomia energética com duração aproximada de um ano, tornando a alimentação do protótipo um aspeto muito importante. A alimentação total do circuito deve ter em conta as necessidades de todos os seus componentes. O Arduino Uno e a placa GSM/GPRS podem receber alimentação de uma fonte até 12V com a particularidade da placa necessitar de até 2A em picos de comunicação. Por outro lado, a válvula necessita de 9V para ser ligada e desligada.

Assim, para alimentar o controlador consideraram-se duas abordagens. Na primeira hipótese pressupõe-se uma única fonte de alimentação partilhada pelo controlador e pela válvula. A principal vantagem desta abordagem é que a substituição das pilhas é um processo mais simples, pois só há um tipo de pilha. A alimentação do controlador e da placa GSM/GPRS tem de ser feita na porta VIN, que converte a tensão para 5V, o que na prática equivale a um desperdício de energia. Outra consequência é que o facto de a mesma fonte ser partilhada faz com que a corrente seja repartida pelos vários componentes (circuito paralelo), podendo ser mais complicado assegurar que todos os componentes recebem a intensidade de corrente desejável.

Na segunda hipótese, que adotámos, o controlador e a placa GSM/GPRS são alimentados a 5V e a válvula a 9V, não sendo necessária a utilização de um conversor de tensão. As vantagens e desvantagens desta alternativa são o oposto da abordagem anterior.

6.4 Programação

A plataforma Arduino disponibiliza um conjunto de bibliotecas que permitem programar o microcontrolador e a interação com diversos componentes através das várias portas digitais e analógicas presentes no controlador. Fazendo uso dessas bibliotecas, criámos um programa com as funcionalidades que consideramos mais prementes: interação com um sensor de temperatura e humidade do ar e com um sensor de luminosidade, comunicação via GPRS e controlo de uma válvula solenóide. Nesta fase não abordámos o adormecer do controlador, nem o agendamento de tarefas associadas a um plano de rega.

6.4.1 Comunicação local com o módulo

Para a comunicação com o servidor remoto utilizando o módulo GPRS é preciso estabelecer a comunicação entre o Arduino Uno e o módulo GSM/GPRS, que designaremos por comunicação local. Esta é feita utilizando o protocolo UART. Na configuração mais simples deste protocolo são usadas apenas as portas RX e TX dos dois dispositivos que estabelecem a comunicação. A única exigência é que a porta RX de um dispositivo deve

ligar-se à porta TX do outro dispositivo e vice-versa, como ilustra a figura 6.8. Adicionalmente os dois dispositivos devem partilhar a mesma terra (GRD).

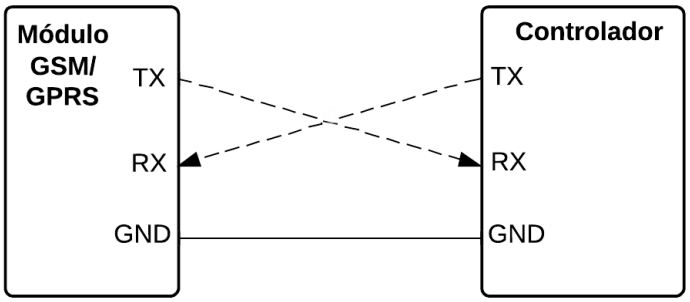


Figura 6.8: Comunicação UART entre o módulo GSM/GPRS e o controlador

Ao usarmos a placa SM5100B acoplada ao Arduino, todas as portas disponibilizadas pelo Arduino passam a estar acessíveis a partir da placa. Significa portanto que as portas RX e TX da placa SM5100B pertencem ao Arduino e não ao módulo GSM/GPRS. Então como estabelecer a comunicação indicada na figura 6.8? Em primeiro lugar, a placa SM5100B apresenta uma configuração especial que mapeia as portas RX e TX do módulo para as portas D2 e D3 da placa, respetivamente, como mostra a figura 6.9.

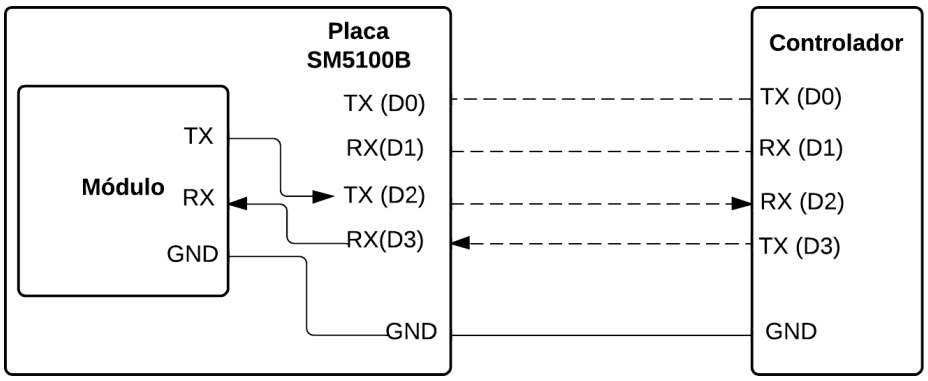


Figura 6.9: Comunicação UART entre a placa GSM/GPRS (com módulo) e o controlador

Mas surge outro problema: o controlador Arduino Uno só aceita comunicações série nas portas D0 e D1. Para mitigar este problema, a plataforma Arduino possui uma biblioteca especial, designada "*Software Serial*", que implementa a comunicação série em qualquer par de portas. Utilizando esta biblioteca podemos programar as portas D2 e D3 do controlador para a comunicação local. A biblioteca "Software Serial" implementa ainda um sistema de memória tampão (*buffer*), que guarda temporariamente os caracteres recebidos do módulo.

O controlo via *software* do módulo GSM/GPRS é feito enviando cadeias de caracteres específicas (comandos) que representam ações a serem executadas (e.g., iniciar ligação remota usando o protocolo TCP). Estes comandos designam-se de comandos AT, largamente utilizados em equipamentos de comunicação. Neste projeto, implementámos uma biblioteca que simplifica a utilização dos comandos AT. A secção 7.1.4 do próximo capítulo contém mais informação sobre esta decisão.

6.4.2 Comunicação remota

Para além da comunicação local é necessário estabelecer uma comunicação remota entre o controlador e o sistema central. Para satisfazer os requisitos do cliente, o nosso objetivo centra-se na conceção de um controlador o mais eficiente possível do ponto de vista energético. Para tal, apresentamos o seguinte modo de funcionamento:

1. O sistema central atua como servidor escutando pedidos de vários controladores;
2. Um controlador acorda e inicia o processo de comunicação, estabelecendo uma ligação ao sistema central através de um nome fixo e bem conhecido;
3. O sistema central controla a conversação enviando sucessivos pedidos de ações e esperando uma resposta do controlador;
4. O controlador termina a ligação quando recebe um comando para este efeito do sistema central.

Os dois primeiros passos permitem que os controladores não estejam em comunicação permanente com o sistema central, poupando energia. Apenas em situações específicas (e.g., antes de se iniciar uma rega) é que o controlador comunica remotamente para atualizar o plano de rega e enviar os dados dos sensores guardados localmente e outras informações relevantes (e.g., estado da bateria). Uma vez que o controlador se regista na rede celular diversas vezes, o seu endereço IP pode variar. Por outro lado, o sistema central está sempre acessível através de um nome conhecido pelos controladores, e quando um controlador quer iniciar uma comunicação utiliza sempre os mesmos parâmetros de ligação. Contudo, ao contrário da arquitetura cliente-servidor tradicional, a partir do momento em que se inicia uma ligação é o sistema central que passa a controlar a conversação enviando sucessivos pedidos de ações e esperando uma resposta do controlador, como descrito no terceiro passo. A comunicação feita é sempre síncrona, sendo que só é enviado um novo pedido após a boa recepção do pedido anterior.

Para efeitos de demonstração ao cliente foi desenvolvida uma aplicação em Java capaz de estabelecer uma comunicação TCP via *sockets* com o controlador e que oferece uma interface simples de texto onde se podem invocar ações no controlador. Esta aplicação pretendia simular as acções do sistema central. Nesta primeira demonstração, o protocolo

de comunicação dita que é o sistema central a terminar a comunicação, enviando para isso a mensagem respetiva. Nessa altura, o controlador fecha a ligação e espera durante um período de tempo até iniciar uma nova ligação. A secção 7.1.1 descreve a otimização deste comportamento na construção do produto final. De seguida, apresenta-se o pseudo-código que resume o funcionamento do controlador.

```
inicia o módulo GSM/GPRS;
inicia ligação TCP com o sistema central;
while o controlador está ligado do
    lê variável de controlo de erro;
    if não há erros then
        bloqueia à espera de um pedido do sistema central;
        lê o pedido do sistema central;
        processa o pedido do sistema central;
        envia uma resposta ao sistema central;
    else
        espera dois minutos;
        inicia o módulo GSM/GPRS;
        inicia ligação TCP com o sistema central;
    end
end
```

Estrutura das mensagens

As mensagens trocadas neste protocolo são também muito simples. Numa primeira fase, definiram-se dois tipos de mensagem: um para pedidos do servidor e outro para respostas do cliente. Associado a cada acção a executar existe um código de mensagem conhecido dos dois lados da comunicação. Os dois tipos de mensagem apresentam o seguinte formato:

- do servidor para o cliente:

```
SERVIDOR_ID # CODIGO_ACAO #
```

- do cliente para o servidor:

```
CONTROLADOR_ID # CODIGO_ACAO_1 : VALOR_ACAO_1 #

... CODIGO_ACAO_n : VALOR_ACAO_n #
```

Tanto do lado do cliente como do lado do servidor as mensagens são sempre construídas como um conjunto de caracteres ASCII.

6.5 Resultados

O primeiro protótipo funcional é constituído por um controlador capaz de comunicar com um servidor remoto e devolver respostas básicas (como por exemplo, leituras de sensores) ou ativar uma válvula solenóide em função dos pedidos do servidor. A demonstração deste protótipo ao cliente veio confirmar que a tecnologia escolhida permite implementar a solução pedida. No entanto, identificámos alguns problemas que analisamos ao longo desta secção e para os quais apresentamos possíveis soluções.

Eficiênia Energética Embora na fase de prototipagem não tenham sido feitos testes relevantes referentes à duração da bateria, a configuração do protótipo revela um conjunto de características que se sabem ser pouco eficientes:

- o controlador Arduino Uno incorpora um microcontrolador extra para conversão de USB para Série não necessário num produto final;
- o controlador Arduino Uno incorpora um conversor de tensão até 12V. Qualquer conversor de tensão é um agente dissipador de energia e quanto maior a tensão de entrada, maior a dissipação. Se o produto final aceitar apenas uma tensão fixa e mais baixa isso pode significar um ganho energético;
- a placa SM5100B é boa para prototipagem, mas apresenta um conjunto de componentes desnecessários no produto final, como a possibilidade de adicionar altifalante e microfone e um conversor de tensão. Uma solução passa por adquirir apenas o módulo GSM/GPRS juntamente com os componentes estritamente necessários;
- o controlador Arduino Uno e a placa SM5100B contêm vários LEDs que são desnecessários.

A configuração do microcontrolador também não está otimizada no protótipo. A velocidade de relógio atual (16MHz) pode ser reduzida e podem ser desativadas funções desnecessárias. A secção 7.3 descreve os esforços feitos neste sentido, na construção do produto final.

Dimensões Comparativamente a outros modelos da família Arduino, o Arduino Uno apresenta dimensões médias. Na verdade o seu tamanho pode ser mais reduzido se não for importante ser compatível com a adição de outras placas e não forem necessárias funcionalidades como a ligação USB. A compatibilidade com mais placas é uma limitação imposta pela utilização da placa SM5100B. Se seguirmos a solução sugerida na secção 6.5 para o módulo, esta limitação desaparece. Por outro lado, num produto final, a maioria dos componentes será do tipo SMD, permitindo obter uma placa com os mesmos componentes, mas de dimensões reduzidas.

Entrada e saída O facto da placa SM5100B utilizar exatamente as mesmas portas que o Arduino Uno facilita a programação do módulo GSM/GPRS, mas a configuração relacionada com a comunicação descrita na secção 6.2.1 obriga a que a utilização das portas D2 e D3 da placa (e consequentemente do Arduino Uno) esteja reservada para comunicação. No entanto, de acordo com as especificações do Arduino Uno (ver a secção 6.1.2), são exatamente essas portas que possibilitam a programação de interrupções face à ocorência de eventos externos (por exemplo, ao ser pressionado um botão). Uma solução para este problema passa por não utilizar a placa acoplada ao Arduino Uno. No entanto, esta solução elimina a vantagem de fácil ligação da placa ao controlador. Uma abordagem preferível é incorporar os dois componentes numa só placa feita à medida com as ligações necessárias.

O microcontrolador ATMEGA328P (presente no Arduino Uno), dispõe de treze portas digitais. Se tivermos em conta que um produto final utiliza várias válvulas solenóides e vários sensores, o número de portas torna-se insuficiente. Para este problema existem pelo menos duas soluções. Na primeira utiliza-se um circuito integrado especial, designado de multiplexador, que permite multiplicar o número de portas digitais (e.g., com três portas digitais, conseguem-se oito portas digitais fornecidas pelo multiplexador). A segunda solução passa por considerar a utilização de um microcontrolador com características diferentes.

Finalmente, a placa SM5100B incorpora o conetor do cartão SIM o que será uma limitação para a instalação de um produto final no terreno. Idealmente, o conetor SIM será independente da placa estando disponível para o utilizador.

Custo de produção O custo de construção do protótipo ronda os 150€. Este valor não é adequado ao requisito do cliente para a produção em massa de um produto final. Um dos componentes que mais encarece o protótipo é a placa SM5100B, com cerca de 65% do custo total, seguido do controlador Arduino Uno, com cerca de 20%. Também aqui uma solução passaria por desenhar uma placa específica contendo apenas os componentes necessários.

Programação O código desenvolvido para o protótipo não contempla várias funcionalidades requeridas para um produto final:

- adormecer o microcontrolador;
- acordar o microcontrolador através de uma interrupção externa (e.g., botão);
- o protocolo de comunicação entre o controlador e o sistema central é muito limitado;
- a comunicação entre o microcontrolador e o módulo GSM/GPRS é lenta e não explora todas as funcionalidades do módulo;

- o desenho de *software* é também simplificado, existindo pouca separação de responsabilidades. Num desenvolvimento final algumas funções podem ser incorporadas numa biblioteca específica melhorando a reutilização de código;
- impossibilidade de guardar e executar um plano de rega periódico.

6.6 Um controlador mais eficiente

Entre os problemas identificados na secção 6.5, a eficiência energética é o mais importante. O desenvolvimento deste controlador não se foca na eficiência energética, mas sim na prototipagem de uso geral. Numa tentativa de desenvolver um protótipo mais eficiente pesquisámos outros controladores da família Arduino com características diferentes, tentando evitar o desenho de uma solução de *hardware* à medida e de raíz.

De entre os vários modelos Arduino, o Arduino Pro Mini, versão 3,3V (figura 6.10), apresenta um conjunto de características muito interessantes no que diz respeito à eficiência energética e às suas dimensões. Apresentamos em baixo um comparativo com o Arduino Uno:

- sem porta USB nem circuito integrado para conversão de USB para Série;
- conetores SPI partilhados com as portas digitais de uso geral;
- igual possibilidade de alimentação até 12V quando utilizada a porta RAW;
- possui o mesmo número de portas digitais e analógicas;
- ocupa menos de metade do tamanho do modelo Arduino Uno;
- é mais barato (cerca de 70% do custo do Arduino Uno);

O microcontrolador usado no Arduino Pro Mini pertence à mesma família que o microcontrolador do Arduino Uno apresentando apenas diferenças na forma como está instalado fisicamente (instalação SMD) e na sua configuração:

- 32 pinos (pacote Plastic Quad Flat Package (TQFP), modelo ATMEGA328P-AU);
- operação a 3,3V;
- velocidade de relógio de 8MHz com oscilador externo.

As características idênticas permitem reutilizar o código já produzido e utilizar as mesmas formas de comunicação com o controlador. As diferenças na configuração acabam também por ser uma vantagem, pois embora a velocidade de processamento seja

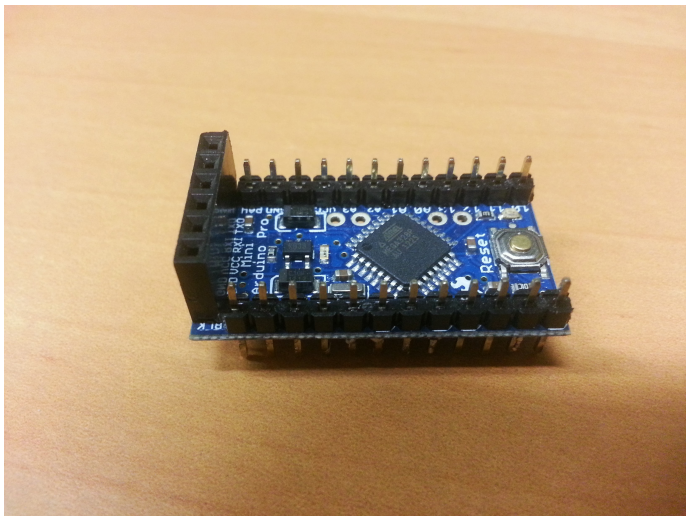


Figura 6.10: Arduino Pro Mini

inferior, não é significativa para os objetivos do projeto e representa uma diminuição no consumo energético.

A utilização do Arduino Pro Mini obriga à aquisição de uma placa extra (FTDI *Basic Breakout* - 3.3V da *SparkFun Electronics* [Spa13]), que permite fazer a conversão de USB para UART, e assim programar o dispositivo através de um computador (ver a figura 6.11). Esta diferença é um melhoramento do ponto de vista energético, pois permite retirar componentes de *hardware* do protótipo, que apenas são necessários durante a fase de programação e que num produto final é feita, em princípio, apenas uma vez.

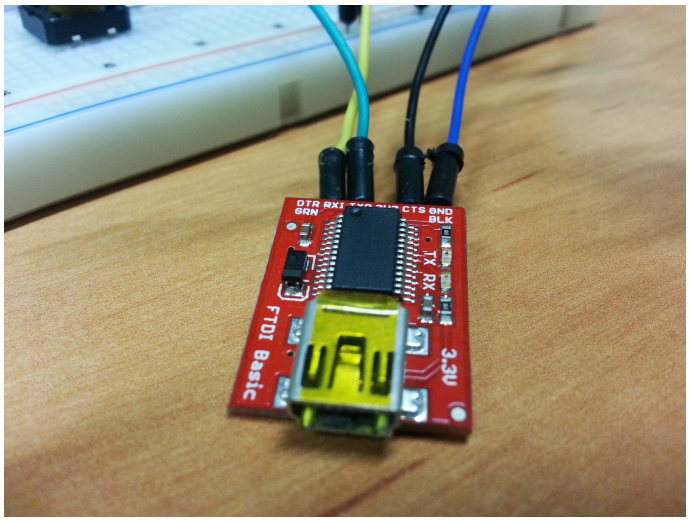


Figura 6.11: Adaptador FTDI

Apesar das experiências com este controlador revelarem melhorias no consumo energético, as suas dimensões reduzidas tornam-no incompatível com a utilização da placa GSM/GPRS. Além disso, não resolve o problema da limitação no número de portas digitais. Não existindo uma solução da família Arduino energeticamente eficiente e ao mesmo tempo

compatível com a placa GSM/GPRS, justifica-se desenvolver uma solução feita à medida. O capítulo 7 descreve a implementação desta solução.

Capítulo 7

Controlador SRI

A plataforma Arduino permitiu-nos criar um protótipo com várias funcionalidades num curto espaço de tempo. No entanto, este protótipo apresenta várias desvantagens e não cumpre ainda todos os requisitos do produto que pretendemos (ver secção 6.5). Apesar de utilizarmos uma placa Arduino diferente, Arduino Pro Mini, chegámos à conclusão que, embora mais eficiente energeticamente, levanta outros problemas tais como a incompatibilidade do *shield* GSM/GPRS com a placa do Arduino Pro Mini (ver secção 6.6). Vimos também que o *shield* GSM/GPRS representa uma percentagem significativa do orçamento total do protótipo que gostaríamos de ver reduzida.

O estudo aprofundado da solução Arduino permitiu-nos concluir que, do ponto de vista energético, não há uma solução pronta a usar (com todas as funcionalidades que pretendemos) e, apesar do foco desta teser ser uma solução de *software*, decidimos ir mais longe e conceber um produto à medida que possa enventualmente ser patenteado e comercializado. Apesar de este ser ainda um trabalho em progressão, o resultado mais importante deste esforço foi o desenho do circuito elétrico do nosso controlador a partir do qual será possível construir o produto final.

Neste capítulo começamos por descrever as funcionalidades de *software* desse produto. O comportamento do controlador é, em princípio, independente do seu *hardware*. Por exemplo, se um dos objetivos do controlador é a comunicação remota, ele deve suportar essa característica independentemente da tecnologia usada para fazer a comunicação (e.g., WI-FI, GPRS, ethernet, etc.). Esta abordagem permitir-nos-á antecipar futuras modificações da instalação do controlador em diferentes cenários e oferece por isso uma grande flexibilidade.

Seguimos depois para a descrição da implementação de *hardware*. À semelhança do processo de desenvolvimento da plataforma de *software*, também a construção de um produto de *hardware* passa por uma fase de desenho. É esse processo que descrevemos também neste capítulo.

7.1 As funcionalidades do controlador SRI

7.1.1 Comportamento

Com a adição de novas funcionalidades no controlador e consequente aumento de complexidade do código, torna-se impreterível repensar a forma como este está programado. A nossa solução implementa uma máquina de estados seguindo uma abordagem idêntica à utilizada na programação da comunicação do sistema central com o controlador (ver secção4.2.8). Esta solução diferencia sete estados de funcionamento com ações distintas.

Iniciar	1	O controlador fornece energia ao módulo GSM/GPRS para que este possa ligar-se e registar-se na rede.
Início de ligação	2	O controlador estabelece uma nova ligação com o sistema central remoto.
Conversação	3	Já existe uma ligação estabelecida entre o controlador e o sistema central e são trocadas uma ou mais mensagens dependendo do contexto (consultar a secção 7.1.4). O controlador pode sair desta fase porque recebeu um comando específico do sistema central, porque excedeu o tempo limite sem receber nenhum pedido novo ou ainda porque deteta que a ligação foi interrompida.
Fim de ligação	4	A ligação é fechada.
Execução	5	Executa a(s) próxima(s) tarefas(s) do plano de rega (consultar a secção 7.1.2).
Adormecer	6	Em primeiro lugar o módulo GSM/GPRS é desligado. Em seguida, o controlador entra num estado de muito baixo consumo. O controlador pode sair deste estado se for ligado manualmente ou para executar uma tarefa do plano de rega.
Recuperação de erro	7	O erro é identificado e o controlador é reiniciado para que possa voltar a um estado de execução normal. Se não conseguir, o controlador adormece. Sempre que possível, o plano de rega instalado continua a ser executado mesmo em caso de erro na comunicação.

O diagrama de fluxo 7.1 ilustra os vários estados e as possibilidades de transição entre os mesmos.

7.1.2 Plano de rega

O plano de rega é o conjunto de eventos de rega que devem ser executados num período de tempo igual a uma semana. Cada evento de rega ativa apenas uma válvula solenóide. A tarefa principal do controlador é armazenar e executar um plano de rega recebido da plataforma central. Podem no entanto existir excepções à execução deste plano consoante as

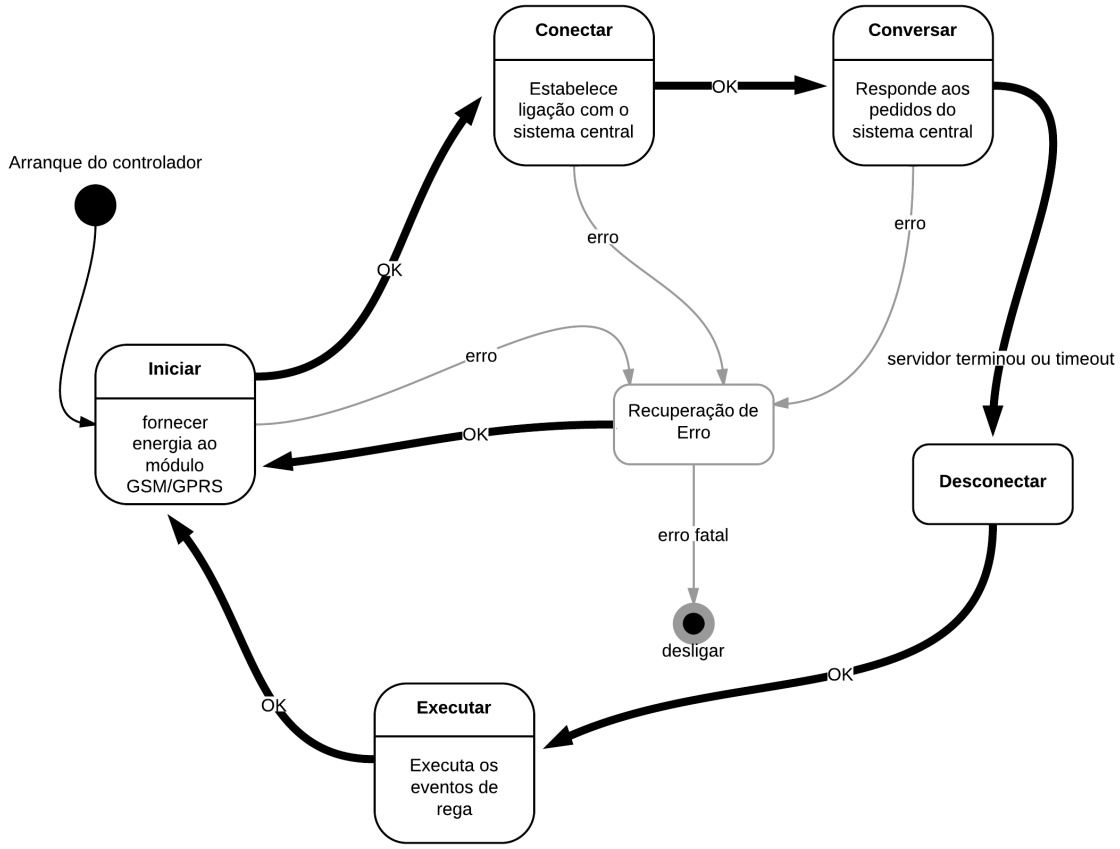


Figura 7.1: Estados do Controlador

condições locais medidas com o auxílio dos sensores. Por exemplo, se houver um período de chuva significativo e inesperado, alguns eventos de rega poderão ser temporariamente suspensos.

Estrutura

Um plano de rega possui tipicamente diversos eventos e cada evento contém informação sobre o dia, a hora, a válvula solenóide a ser ativada, a duração da rega, etc. É fundamental otimizar a forma como esta informação é armazenada. Para além das óbvias limitações de memória do microcontrolador existem ainda limitações relacionadas com o tamanho máximo do *buffer* da biblioteca de comunicação local "SoftwareSerial" e também da própria comunicação GPRS.

De acordo com a arquitetura do microcontrolador, o mínimo de informação que pode ser guardada em memória corresponde a 1 byte (8 bits). A nossa implementação representa os campos associados a um evento do plano da seguinte forma:

Dia da semana	Cada semana tem 7 dias e portanto precisamos de 3 *bits* (com 3 *bits* é possível representar 8 números)
Hora de início	Um dia tem 24 horas, logo são necessários 5 *bits* (embora aqui haja algum desperdício, 4 *bits* não seriam suficientes).
Minuto de início	Cada hora tem 60 minutos, mas podemos simplificar em múltiplos de 5. O número máximo é o 11, logo são necessários 4 *bits*.
Hora de duração	Tipicamente um evento de rega não dura mais de 7 horas, logo são necessários 3 *bits*.
Minuto de duração	Igual ao minuto de início (4 *bits*).
Número da válvula	O número de válvulas pode variar mas vamos assumir que o controlador opera no máximo com 6 válvulas. Logo são necessários 3 *bits*.

Se utilizássemos um *byte* para guardar cada um destes campos estaríamos a desperdiçar espaço em memória. Por exemplo, para guardar sete dias da semana necessitamos de apenas três *bits* o que significa que estaríamos a desperdiçar cinco *bits*. Assim, estruturamos todas as informações de um evento em vinte e dois *bits* ($3 + 5 + 4 + 3 + 4 + 3 = 22$). Para guardar vinte e dois *bits* precisamos de 3 *bytes* ($3 \times 8 = 24$) e ainda nos sobram 2 *bits*. O campo que faz mais sentido aumentar a capacidade de armazenamento é o número da válvula ficando assim com 5 *bits* disponíveis o que permite ter um controlador capaz de trabalhar com 32 válvulas. Em baixo apresenta-se a estrutura final para um evento considerando a ordenação de *bytes* "*big endian*".

23 22 21 20 19	18 17 16	15 14 13 12 11	10 9 8	7 6 5 4	3 2 1 0
Válvula	Dia	Hora	Dur_Hora	Minuto	Dur_Min
1 byte		1 byte		1 byte	

Uma vez que o valor zero pode assinalar o fim de uma mensagem para o módulo GSM/GPRS, esse valor nunca é usado para nenhum dos campos do evento. No final deste documento descrevem-se os valores que podem ser utilizados para cada campo e qual o seu signficiado (ver Apêndice C.2). Apresentamos em baixo dois exemplos de eventos codificados.

Exemplo 1. *Um evento para regar a válvula nº3, aos Domingos, a começar às 17h20m e com uma duração de 2h25m será representado por três inteiros de 8 bits: 25, 138 e 69.*

23 22 21 20 19	18 17 16	15 14 13 12 11	10 9 8	7 6 5 4	3 2 1 0
Válvula	Dia	Hora	Dur_Hora	Minuto	Dur_Min
00011	001	10001	010	0100	0101_Min
3	Domingo	17	2	4x5min=20	5x5min=25
25		138		69	

Exemplo 2. *Um evento para regar a válvula nº5, às 4ª-Feiras, a começar às 0h55m e com uma duração de 1h30m será representado por três inteiros de 8 bits: 41, 193 e 182.*

23 22 21 20 19	18 17 16	15 14 13 12 11	10 9 8	7 6 5 4	3 2 1 0
Válvula	Dia	Hora	Dur_Hora	Minuto	Dur_Min
00101	100	11000	001	1011	0110
5	4ª-Feira	24 (0h)	1	11x5min=55	6x5min=30
44		193		182	

Execução

Quando o controlador possui um plano de rega guardado, a execução do plano é independente da comunicação com a plataforma. Isso permite que o controlador execute o plano de rega mesmo na eventualidade de não conseguir comunicar com a plataforma central. Para evitar um desperdício de energia, o controlador está adormecido o máximo de tempo possível e permite agendar eventos para voltar ao estado ativo. O agendamento destes eventos é ditado pelas tarefas existentes no plano de rega. Assim, antes de adormecer, o controlador procura no plano de rega qual a próxima data em que deve executar uma tarefa de rega e agenda esse evento. Quando volta do estado de repouso, as tarefas são executadas e o processo repete-se avançando no tempo. Este ciclo tem a duração de uma semana, ao fim da qual o controlador volta a ler a primeira tarefa do plano de rega. Como referido na secção 7.1.2 as tarefas do plano não contêm informação sobre o dia do ano, mas sim sobre o dia da semana. O controlador é responsável por agendar os próximos eventos no tempo em função da informação lida em cada tarefa.

7.1.3 Regulação e monitorização da rega

Para além de ser possível agendar um plano de rega, o controlador faz ajustes automáticos com base na medição de diversos parâmetros meteorológicos. Esta medição é feita a partir da informação dos sensores ou através de informção recebida da plataforma de gestão (e.g., informação de uma estação meteorológica).

Um exemplo desta regulação automática é o caso em que o controlador tem um plano de rega atribuído mas que uma, ou mais, ações de rega são canceladas, pois houve um período de chuva que não estava previsto. A monitorização feita pelos sensores será ainda enviada regularmente para a plataforma central de forma a que o operador do sistema seja alertado e possa agir convenientemente.

Será com base na medição regular deste tipo de valores que, ao longo do tempo, as alterações ao plano de rega permitirão cada vez mais a sua aproximação às necessidades reais de cada jardim em particular e diminuir o consumo de água.

7.1.4 Comunicação com a plataforma central

A comunicação remota com a plataforma central de gestão é um dos requisitos principais do controlador. Durante o desenvolvimento do protótipo inicial, vimos na secção 6.4.1 que foi utilizada a biblioteca "*Software Serial*" para comunicar com o módulo GSM/GPRS e que essa comunicação utilizava um protocolo com comandos específicos (comandos AT) que eram interpretados pelo módulo. Tal como dissemos no início deste capítulo, a solução de *software* responsável pela comunicação deve ser independente da tecnologia utilizada. Por essa razão, desenvolvemos uma biblioteca de comunicação para abstrair a comunicação com o módulo GSM/GPRS distinguindo as operações que dizem respeito ao funcionamento básico do controlador (e.g., execução do plano de rega) e aquelas que são específicas da comunicação (e.g., receber um novo plano da plataforma central). Esta biblioteca está desenvolvida na linguagem C++ e é constituída por uma interface e uma classe de implementação. Para além de permitir diferentes módulos GSM/GPRS, esta biblioteca facilitará no futuro a adaptação a diferentes tecnologias, como por exemplo a tecnologia sem fios WI-FI, caso haja necessidade. Como complemento desta biblioteca, criámos ainda um Tipo de Dados Abstrato (TDA) que permite descodificar e armazenar a informação das mensagens recebidas da plataforma central.

7.1.5 Protocolo de comunicação

O protocolo de comunicação do controlador SRI utiliza a mesma arquitetura do protótipo (ver secção 6.4.2). A estrutura das mensagens trocadas foi otimizada satisfazendo o número maior de funcionalidades. À semelhança do protótipo, as mensagens trocadas nos dois sentidos da comunicação utilizam códigos que representam os pedidos a serem executados no controlador e os valores da resposta a esses pedidos, mas no caso específico das mensagens de resposta do controlador, elas possuem agora um código especial que fornece informação sobre o estado do pedido. Durante a execução do pedido existem três estados distintos:

- OK: o pedido foi executado com sucesso;
- NOTOK: não foi possível executar o pedido porque ocorreu um erro;
- RESEND: foi excedido o tempo de espera por um pedido do servidor e o controlador pede que seja enviado de novo o mesmo pedido.

Importa referir que o inteiro "0" nunca é utilizado em nenhuma das mensagens pois a sua interpretação pode ter significados diferentes dependendo da implementação nos diferentes componentes de *hardware*. Para todos os casos em que é preciso referenciar o valor zero é utilizado o inteiro "255".

De seguida descrevemos com mais detalhe as mensagens enviadas para o controlador e as mensagens enviadas para a plataforma central. Como complemento desta descrição,

a secção C.3 em apêndice resume o cenário típico de comunicação entre o sistema central e o controlador e a descrição detalhada dos códigos pode ser consultada também em apêndice na secção C.1.

Mensagens enviadas para o controlador

Cada mensagem trocada entre o sistema central e o controlador contém, tal como anteriormente, a identificação do sistema central e um código associado a uma ação específica. Contudo, cada ação pode agora ter um conjunto de dados associados, como são os casos em que é enviado um novo plano de rega ou é alterado o estado de uma válvula. Para diminuir o tamanho, as mensagens enviadas para o controlador passaram a ser estruturadas como uma cadeia de *bytes*. Esta alteração tornou-se especialmente necessária quando se definiu a estrutura de dados que representa um evento do plano de rega no controlador (ver secção 7.1.2). Além desta alteração, o identificador do servidor foi simplificado para um único número inteiro. Quanto à estrutura de dados, como o tamanho é variável, é necesário incluir na mensagem um campo extra, que define o tamanho desta em *bytes*.

Formato 1. *Estrutura da mensagem enviada do sistema central para o controlador.*

```
SERVIDOR_ID CODIGO_PEDIDO N_BYTES DADOS
```

De seguida apresentamos um exemplo onde é comparada a estrutura das mensagens para as duas abordagens: vetor de caracteres e vetor de *bytes*. O plano aqui representado possui os dois eventos descritos anteriormente na secção 7.1.2.

Exemplo 3. *O servidor com o identificador "SRV43" ("43" na nova abordagem) envia um pedido para o controlador cuja acção é executar um novo plano de rega (corresponde ao código "102"). O plano é constituído por dois eventos: "válvula nº3, Domingo, começa às 17h20m, duração de 2h25m" e "válvula nº5, 4ª-Feira, começa às 0h55m, duração de 1h30m".*

Abordagem antiga (caracteres ASCII)

ID	Código	Dados
#SRV43#	102#	25#138#69#44#193#182#
7 bytes	4 bytes	21 bytes

Total: 32 bytes

Abordagem nova (*bytes*)

ID	Código	Tamanho	Dados
43	102	6	25 138 69 44 193 182
1 byte	1 byte	1 byte	6 bytes

Total: 9 bytes

Como se pode comprovar pelo exemplo, a utilização de um vetor de *bytes* (sem associação ASCII) é claramente preferível se quisermos poupar espaço, especialmente nos casos de pedidos em que é enviada uma grande quantidade de dados (e.g., eventos do plano).

Mensagens enviadas para o sistema central

A estrutura das mensagens de resposta ao sistema central é a mesma utilizada na construção do protótipo (ver secção 6.4.2) mas adicionámos um campo extra que permite controlar o estado de um pedido (ver secção C.1). A secção 8.2 justifica algumas alterações que poderão ser implementadas no futuro.

Formato 2. *Estrutura da mensagem enviada do controlador para o sistema central. Um pedido do servidor pode corresponder à execução de várias ações distintas (e.g., leitura dos valores de todos os sensores) e por isso a resposta do controlador pode conter vários valores. Para além dos valores das ações, cada resposta tem sempre um campo especial que informa o servidor sobre o estado do pedido.*

Código identificador	`CODIGO_ID : CONTROLADOR_ID #`
Código estado	`CODIGO_ESTADO : VALOR_ESTADO #`

Ações	`CODIGO_ACAO_1 : VALOR_ACAO_1 #`
	`CODIGO_ACAO_2 : VALOR_ACAO_2 #`
	`...`
	`CODIGO_ACAO_n : VALOR_ACAO_n #`

7.2 Componentes de *hardware*

A implementação das funcionalidades do controlador SRI é suportada por um conjunto de componentes de *hardware* interligados entre si formando um circuito eletrónico. O circuito que desenhámos permitirá a impressão de uma PCI preparada para conectar os componentes internos (microcontrolador, módulo GSM/GPRS, resistências, etc.) e os componentes externos (sensores, válvulas, etc). Por razões de ordem técnica o circuito desenhado utiliza apenas componentes capazes de soldar manualmente. Numa fase posterior este circuito poderá ser convertido facilmente para uma versão do tipo SMD. A figura 7.2 apresenta uma visão de alto nível do circuito final. De seguida descrevemos cada um dos componentes que devem integrar o controlador SRI.

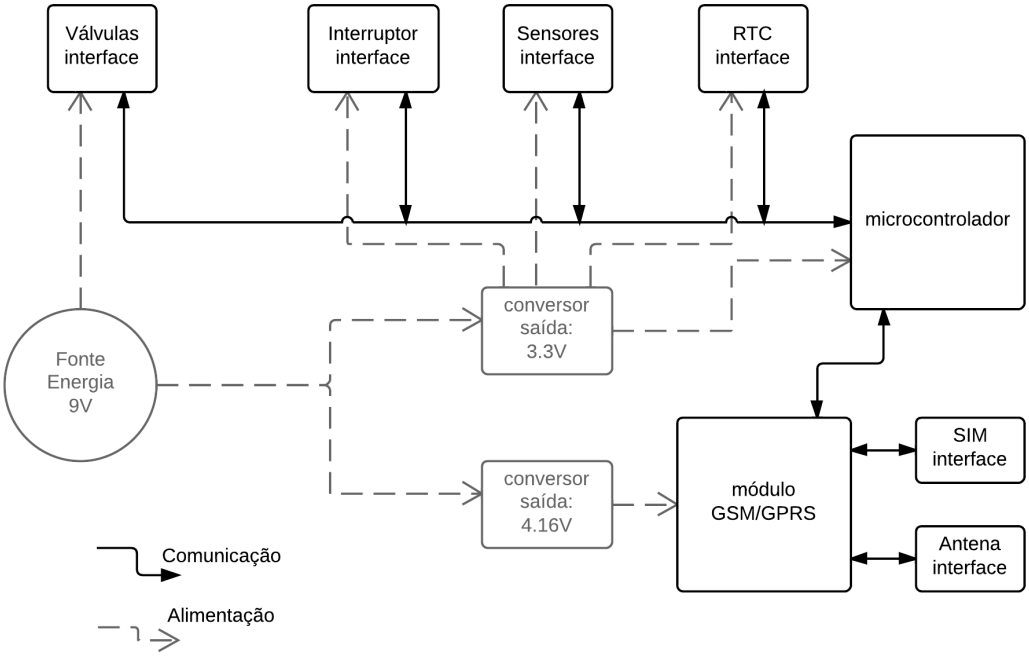


Figura 7.2: Visão alto nível do circuito final

7.2.1 Microcontrolador

A utilização do microcontrolador ATMEGA328P durante a fase de prototipagem revelou duas grandes limitações: o número de portas digitais de E/S era insuficiente para o número de sensores e válvulas que era necessário controlar e o tamanho da memória podia ser uma limitação para guardar planos com um número grande de eventos. Ponderámos por isso a utilização de um microcontrolador diferente. No entanto, era importante que o novo microcontrolador apresentasse algumas características idênticas permitindo acima

de tudo reutilizar todo o código já desenvolvido (incluindo aqui as bibliotecas da plataforma Arduino). A nossa escolha incidiu no microcontrolador ATMEGA644 em vez do ATMEGA328P pelas seguintes razões:

- tem o dobro da capacidade de RAM (4 KBytes) e EEPROM (2 KBytes);
- mais pinos (40 pinos);
- tensão de entrada idêntica (entre 2,7V e 5,5V);
- disponibiliza os mesmos protocolos de comunicação;
- API para programação idêntica;
- dimensões adequadas para instalação manual (pacote DIP).

O controlador SRI possui um conector DIP de 40 pinos para encaixar o chip ATMEGA644. A figura 7.3 ilustra o novo microcontrolador.

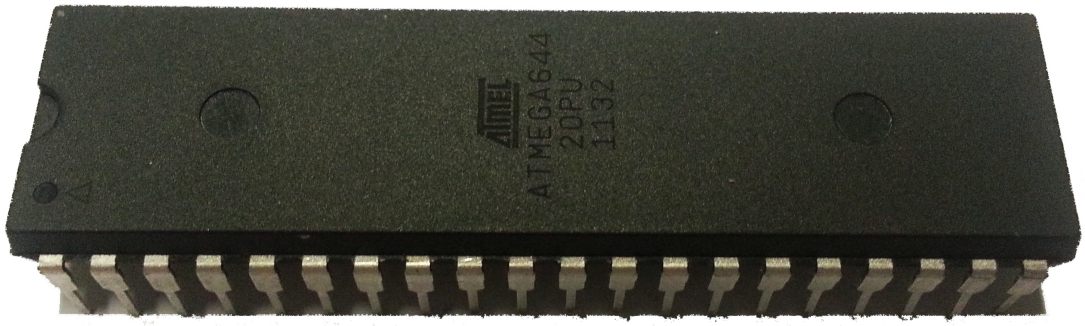


Figura 7.3: Microcontrolador ATMEGA644

Mapeamento dos pinos

O microcontrolador ATMEGA644 é um circuito integrado e por isso incorpora vários componentes num único pacote. Para usufruir das funcionalidades do microcontrolador este possui um conjunto de pinos com funções distintas e cada pino tem um identificador único permitindo modificar o seu valor através de *software*. A programação de baixo nível do microcontrolador pode ser abstraída com uma camada de *software* intermédia que expõe uma interface simples de utilizar fazendo o mapeamento dos identificadores dos pinos para um número correspondente a uma porta digital ou analógica. A plataforma Arduino já possui um conjunto de bibliotecas para esse efeito mas que obviamente tem pequenas variações consoante o microcontrolador alvo.

Apesar da programação e funcionalidades do chip ATMEGA644 e do chip ATMEGA328P serem idênticas há uma diferença grande no mapeamento dos pinos. Para além do ATMEGA644 possuir mais pinos, a função de cada pino não tem uma correlação com o pino correspondente no ATMEGA328P. A plataforma Arduino não possui nenhuma extensão para o chip ATMEGA644 mas, à data de pesquisa, encontrámos um projeto relevante, o Sanguino [Hoe08], cujo controlador utiliza também um microcontrolador da família ATMEGA644.

Todos os pinos permitem uma ligação digital mas alguns têm funções extra. A figura 7.4 ilustra o mapeamento dos pinos do ATMEGA644 para o controlador SRI.

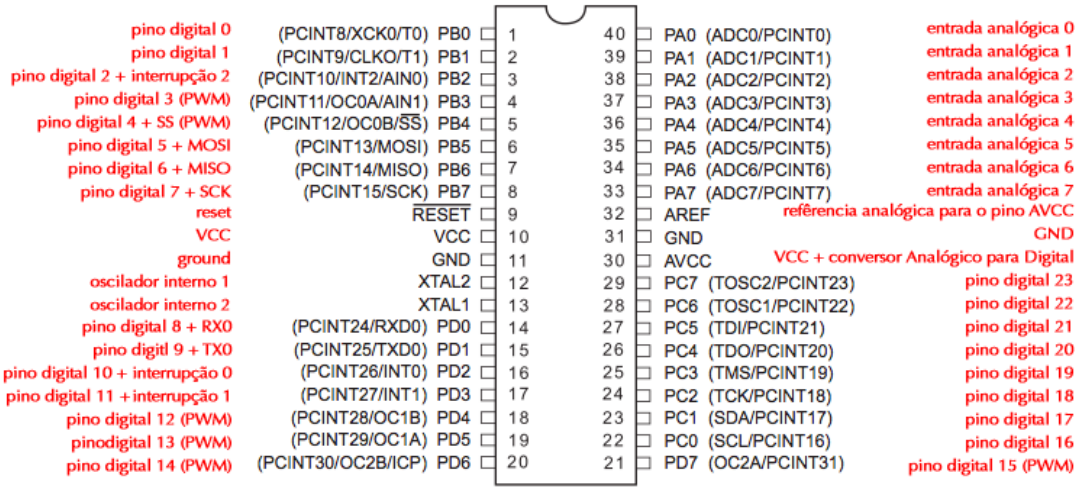


Figura 7.4: Mapeamento de pinos ATMEGA644

Método de programação do microcontrolador

No caso dos microcontroladores Atmel AVR (dos quais faz parte a família ATMEGA), as instruções executadas pelo microprocessador são guardadas na memória FLASH, uma memória não volátil e reprogramável. Para reescrever o conteúdo da memória existem duas possibilidades: via hardware e via software.

Quando o microcontrolador recebe energia ou é feito *reset*, ele executa automaticamente as instruções guardadas na memória. Se quisermos reescrever o conteúdo da memória é necessário colocar o microcontrolador num estado especial. Isso é possível utilizando um dispositivo auxiliar, designado de *Programador*, que é ligado ao microcontrolador num conjunto de pinos específicos. A modificação do estado destes pinos permite fazer *reset* ao microcontrolador, suspender a execução de instruções da memória FLASH e enviar novas instruções que são escritas na memória. Este procedimento utiliza o protocolo SPI e os pinos utilizados são os correspondentes a esse protocolo, tal como descrito anteriormente na secção 6.1.2 durante a descrição do protótipo.

Embora este seja um processo com várias limitações (e.g., mais hardware, programação tem que ser local) ele tem que ser feito pelo menos a primeira vez em que é utilizado o microcontrolador pois a memória não tem ainda nenhum conteúdo.

A figura 7.5 ilustra a reprogramação do microcontrolador ATMEGA644 recorrendo a um Arduino Uno como dispositivo programador externo. A configuração mais básica deste microcontrolador apenas necessita das ligações "VCC" (tensão positiva '+') e "GRD" (terra '-'). Para a programação utilizando o protocolo SPI são necessárias as ligações "MOSI", "MISO" e "SCK" e ainda a ligação "RST" que permite fazer *reset* ao microcontrolador. Para mais informações sobre este protocolo, consultar a tabela 6.1 na secção 6.1.2. É comum utilizar um condensador perto dos pinos "VCC" e "GRD" para reduzir o ruído eletrónico.

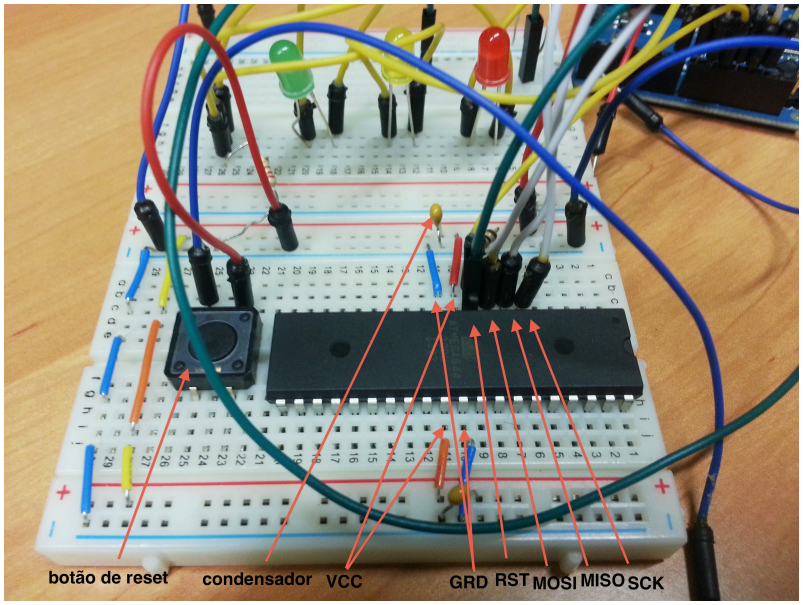


Figura 7.5: Programação do microcontrolador ATMEGA644 com utilização de um arduino UNO como programador

O ATMEGA644 separa logicamente a memória FLASH em duas secções independentes - "aplicação" e "*bootloader*" - e que podem ser reescritas por *software*, ou seja, o microcontrolador executa instruções da memória que reescrevem a própria memória. A principal diferença do *bootloader* em comparação com a aplicação principal é precisamente o sítio da memória onde são escritas as instruções. Sempre que é feito *reset* ao microcontrolador, as instruções da zona de memória "*bootloader*" são executadas em primeiro lugar. Por outro lado, se o microcontrolador iniciar sem ser feito *reset* o controlo do programa salta automaticamente para a memória da aplicação e as instruções do *bootloader* são ignoradas.

A utilização de um *bootloader* é uma alternativa muito poderosa à reprogramação do microcontrolador via *hardware* pois permite que o programador instale novas versões da aplicação sem necessidade de utilizar um programador externo. Uma configuração

comum consiste em escrever um *bootloader* que escuta numa porta de comunicação (e.g., UART, se for comunicação série) por novas instruções. Se forem recebidas novas instruções, o *bootloader* reescreve a zona de memória da aplicação com essas instruções. Se não forem recebidas instruções, ao fim de um certo período de tempo, a execução salta para a zona de memória da aplicação. Em configurações mais avançadas pode até acontecer que o *bootloader* atual reescreve a zona de memória do *bootloader* substituindo o *bootloader* existente por um novo. Neste exemplo concreto é possível comunicar entre o microcontrolador e um computador com um cabo série diretamente. Com esse mesmo tipo de comunicação, pode ser possível, noutro cenário de exemplo, ligar o microcontrolador a um módulo GSM/GPRS e através de comunicação série enviar comandos AT que permitam interagir com um dispositivo remotamente via TCP.

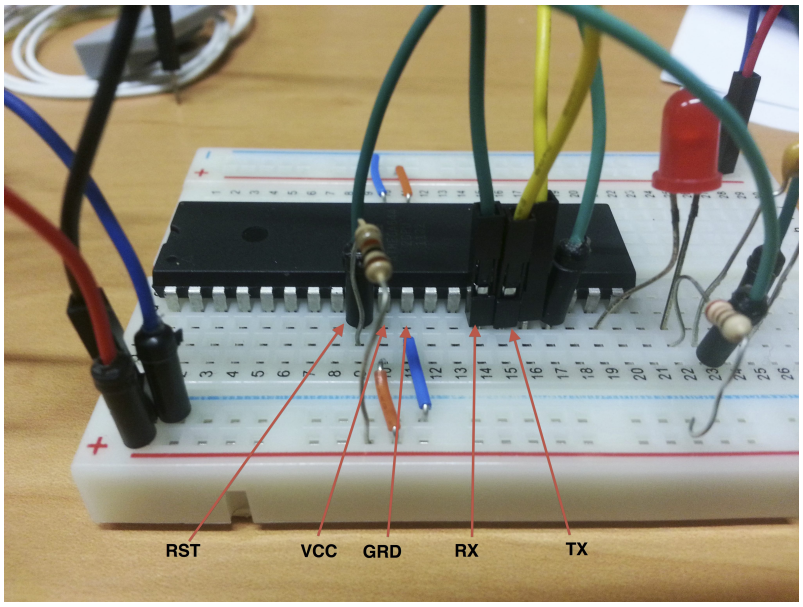


Figura 7.6: Programação do microcontrolador ATMEGA644 via comunicação série

A figura 7.6 ilustra a utilização de um microcontrolador ATMEGA644 que já tem um *bootloader* instalado e que utiliza as portas série (RX - recetor e TX - transmissor) para comunicar com outros dispositivos. Também nesta configuração é utilizada a ligação de *reset*.

Importa ainda referir que quando é feita a programação via *hardware* é possível mudar algumas propriedades do microcontrolador. Entre elas destacamos o tamanho de cada uma das zonas de memória, se essas zonas são apenas de leitura/escrita, qual a frequência de relógio do CPU e qual a fonte de relógio que deve ser utilizada (oscilador interno ou oscilador externo). Estas configurações têm impacto direto no desempenho e no consumo energético do microprocessador. A secção 7.3 sobre poupança energética justifica algumas das características anteriores relacionadas com a diminuição do consumo energético e a secção 8.1 sobre trabalho futuro descreve outras possibilidades de configuração do

bootloader com vista à reprogramação remota. Finalmente, na secção E em apêndice podem ser encontradas mais informações sobre a configuração do *bootloader* instalado no controlador SRI.

RTC auxiliar

O controlador SRI possui um Real Time Clock (RTC) incorporado (ver figura 7.7). Este componente permite agendar uma interrupção de hardware no tempo e sem ele o microcontrolador não poderia adormecer mais de oito segundos (ver secção 7.3). O RTC pode ser alimentado através do microntrolador ou através de uma pilha de pequenas dimensões.

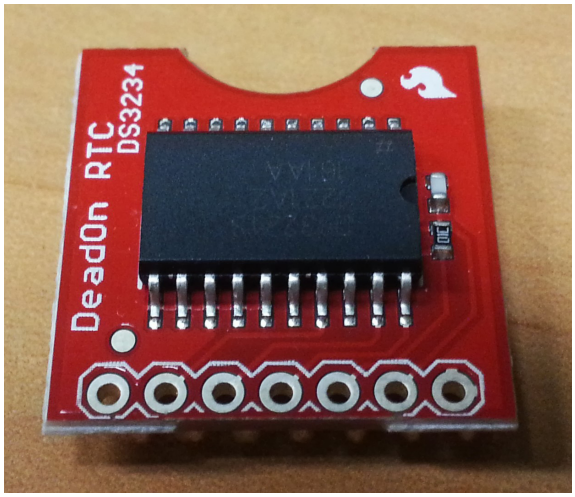


Figura 7.7: RTC DS3234 com placa integrada

7.2.2 Módulo GSM/GPRS

A utilização do *shield* SM5100B na construção do protótipo revelou três grandes desvantagens:

- Baixa eficiência energética (secção 6.5);
- Preço elevado (secção 6.5);
- Incompatibilidade física (6.6).

Para além destas desvantagens, as secções respetivas apontavam uma solução comum: adquirir um módulo GSM/GPRS específico e desenhar um circuito dedicado com os componentes estritamente necessários. Fizemos por isso uma pesquisa no mercado para adquirir um módulo GSM/GPRS com as seguintes características: baixo custo, alto rendimento energético, capaz de ser soldado manualmente, boa documentação e programação utilizando comandos AT.

Optámos então pelo modelo M10 da Quectel [Que13]. Apesar desta empresa estar filiada na China, possui parceiros em diversos pontos do mundo sendo Espanha o país representado mais próximo de Portugal. Cada módulo tinha à data de aquisição o preço de 17€. Embora para integrar este módulo no circuito final sejam necessários mais alguns componentes, o preço final é incomparavelmente reduzido com a solução anterior. A figura 7.8 mostra lado a lado os dois módulos diferentes.

Figura 7.8: Comparação entre os módulos M10 e SM5100B

Preparação do módulo para utilização

Ao contrário do módulo SM5100B que era ligado a um único conector na placa GSM/GPRS (*shield*), o módulo M10 possui uma configuração diferente com vários pinos na borda de cada uma das faces. A ligação deve ser feita individualmente para cada pino o que na prática signfica soldar um pequeno fio a cada um deles. A utilização de todos estes conectores não é obrigatória e depende das funcionalidades que se pretende tirar partido do módulo. A figura 7.9 ilustra um módulo M10 em fase de preparação.

O manual de documentação do módulo M10 possui todas as informações necessárias para preparar este dispositivo para utilização. A figura 7.10 esquematiza o módulo M10 indicando para cada pino a sua função. Face às necessidades para o nosso produto, apenas alguns pinos são necessários (marcas a vermelho):

- VBAT, GRD, AGRD (alimentação);
- RXD, TXD, DTR, RTS, CTS, RI (comunicação série UART);
- RF_ANT (antena);

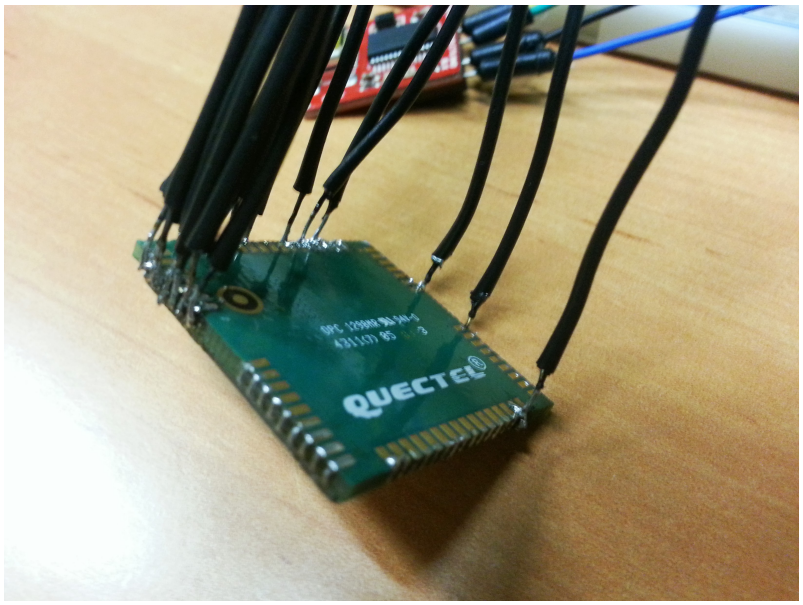


Figura 7.9: Módulo Quectel M10 em fase de preparação

- SIM_VDD, SIM_DATA, SIM_CLK, SIM_RST (cartão SIM de 5 pinos);
- STATUS (indicador do estado operacional do módulo);
- PWERKEY (botão de alimentação).

7.2.3 Componentes externos

Para além dos componentes integrados na placa principal, o controlador SRI oferece suporte à ligação de componentes externos. Estes são componentes que não poderão ficar dentro da caixa envolvente da placa devido à sua função. É o caso dos vários sensores, das válvulas ou do conetor para o cartão SIM. Embora a conceção da caixa do controlador SRI saia fora do âmbito desta tese, o desenho da placa já deve ter em conta quais as portas externas e tornar esse processo o mais simples possível. Por exemplo, no caso das válvulas em que é necessário um circuito dedicado (com transístor e díodo), esse circuito é incluído na placa sendo que a ligação da válvula à caixa só necessita das duas ligações convencionais (positivo e *ground*). Prevendo alterações futuras às necessidades de controlo da rega, o desenho da nossa placa deixa ainda algumas portas livres sem nenhuma função específica atribuída.

A tabela em baixo descreve as portas que têm uma função bem conhecida:

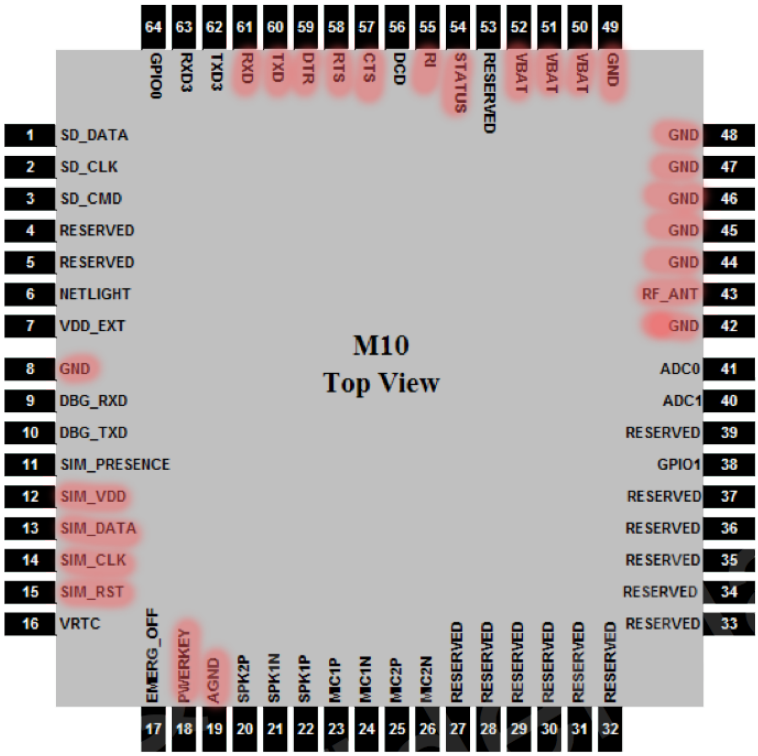


Figura 7.10: Significado dos pinos do módulo M10

Identificador da porta	**Função da porta**	**Descrição dos conectores**
P1	Temperatura e Humidade do ar	(+) positivo; (-) ground; (d) dados
P2	Humidade do solo	(+) positivo; (-) ground; (d) dados
P3	Luminosidade	(+) positivo; (-) ground
P4	Caudalímetro	(+) positivo; (-) ground; (d) dados
P5	Ativação manual (botão de pressão)	(+) positivo; (-) ground; (d) dados
P6	Conetor para cartão SIM	(+) positivo; (-) ground; (d) dados
P7-14	Válvulas (1-8)	(+) positivo; (-) ground;
P15	Alimentação (9V)	(+) positivo; (-) ground;

7.2.4 Alimentação

No capítulo anterior onde falámos sobre o protótipo desenvolvido referimos duas abordagens possíveis para a alimentação dos componentes do circuito (ver secção 6.3). Concluímos que a solução mais eficiente passava por ter mais do que uma fonte de alimentação e foi essa a decisão tomada na implementação do protótipo.

Contudo, embora a eficiência energética seja um dos requisitos principais a compatibilidade e facilidade de instalação do controlador SRI em sistemas de rega já existen-

tes também é uma preocupação fundamental. Face a este cenário, consideramos mais plausível a hipótese de ter uma única fonte de alimentação e em contrapartida utilizar conversores de tensão mais eficientes. A nossa pesquisa revelou que existem várias alternativas para fazer a conversão de tensão. Destacamos aqui as três soluções mais comuns:

- DC/DC tradicional:
- SMPS (*switcher*: It is a switchmode regulator which means that it is very efficient at converting the voltages, and will use a lot less input current to provide the same output current.
- LDO:

Este é um aspeto que ainda se encontra em investigação à data de conclusão desta tese. A figura 7.11 ilustra a configuração da alimentação do circuito do controlador SRI referida anteriormente.

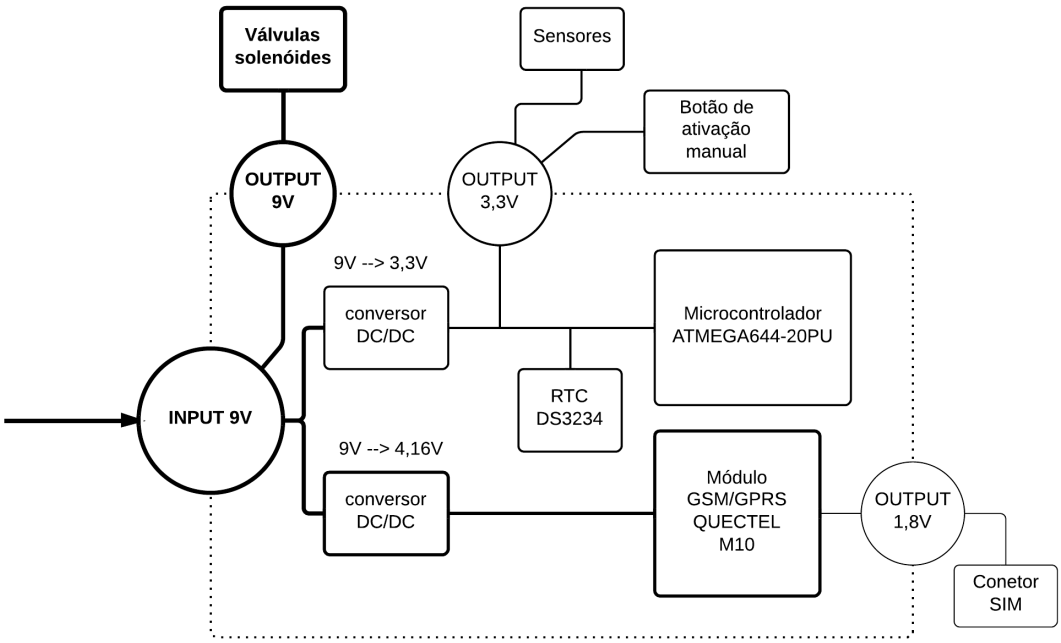


Figura 7.11: Alimentação do controlador SRI

7.3 Poupança Energética

A poupança energética foi identificada como uma preocupação crucial ao longo de vários cenários abordados até agora relacionados com a construção do controlador SRI. Esta secção pretende resumir os esforços feitos no sentido de tornar o controlador SRI num

dispositivo com alta eficiência energética incluindo algumas decisões já referidas e acrescentando outras decisões também importantes.

De seguida descrevemos as medidas que acreditamos terem mais impacto na redução do consumo energético do controlador SRI.

Circuito feito à medida Tal como apontado na secção 6.5 do capítulo anterior, a decisão de desenhar um circuito à medida elimina os componentes desnecessários e que representam um desperdício energético.

Comportamento do controlador O consumo energético do microcontrolador ATMEGA644 varia muito com a carga de instruções que são executadas pelo microprocessador. Assim, a aplicação desenvolvida e que é executada pelo microcontrolador também tem impacto no consumo indiretamente. Como foi descrito na secção 7.1.1, o controlador SRI passa uma grande parte do tempo inativo, altura em que aguarda por executar uma nova tarefa do plano de rega. O comportamento do controlador SRI foi pensado com vista à redução máxima do trabalho executado:

- Regra geral o controlador está adormecido;
- O controlador só comunica com o servidor antes de ser iniciada uma nova rega ou quando é ativado o botão de ativação manual;
- Como alternativa ao ponto anterior e caso seja mesmo necessário, durante uma rega o controlador pode adormecer e acordar periodicamente para saber se deve parar antes do tempo estipulado no plano de rega;

Durante o período de tempo em que o controlador está inativo, o ideal seria desligar o controlador por completo. Porém, desligando o microcontrolador não há maneira deste voltar a ligar-se autonomamente. A estratégia passa então por colocá-lo num estado de execução que lhe permita consumir o mínimo de energia possível (*sleep*). Como referido na secção 7.2.1 do capítulo anterior, o microcontrolador possui vários modos de redução de atividade cada um dos quais com características e resultados no consumo diferentes. Estes modos diferenciam-se essencialmente pela quantidade de funcionalidades que são desativadas no *hardware* do microcontrolador. O estado que permite obter um consumo mais baixo é aquele que desativa mais funcionalidades e que demora mais tempo a voltar ao estado normal de execução.

Dependendo dos requisitos de cada produto, o tipo de *sleep* varia pois alguns modos desativam funcionalidades que são necessárias. Felizmente esse não é o caso do controlador SRI. Durante o período de inativação do controlador SRI ele pode estar completamente adormecido o que nos permite utilizar o modo *sleep* mais eficiente, o "Power-down". O único requisito é que seja possível "acordar" o microcontrolador de duas maneiras: ativando um temporizador e através de uma operação manual (e.g., pressionando

um botão). O modo "Power-down" permite reativar o microcontrolador com interrupções externas (que alteram o estado dos pinos) e ativando o temporizador interno, o que aparentemente vai de encontro aos dois requisitos levantados. Contudo, existe uma limitação relacionada com o temporizador interno do microcontrolador ATMEGA644 que não consegue contar um período de tempo superior a oito segundos. Se quisermos "adormecer" o controlador por um período maior de tempo então é necessário utilizar um dispositivo auxiliar, um RTC, já referido na secção 7.2.1.

Configuração do microcontrolador Como já referido na secção 7.2.1, o microcontrolador ATMEGA644 pode ter diferentes comportamentos durante a sua execução. Esta configuração é feita mudando o conteúdo de uma zona de memória especial e que é vulgarmente referida como *fuse bits*. Esta zona de memória é completamente independente das três memórias principais do microcontrolador (SRAM, EEPROM e Flash) e não é alterada quando o conteúdo do microcontrolador é apagado (processo designado de *chip erase*). Na secção E em anexo pode ser encontrada mais informação sobre a configuração do microcontrolador visando a poupança energética.

Configuração do módulo GSM/GPRS Uma das razões que nos loevou a escolher o módulo QUECTEL M10 é o facto de este ser um dispositivo com elevada eficiência energética. Ainda assim, existem algumas medidas que podem ser tomadas para otimizar o seu rendimento mas que saem do âmbito desta tese:

- A intensidade de corrente fornecida pela bateria pode ser reduzida se forem colocados condensadores próximos do pino VBAT;
- A classe utilizada na comunicação GPRS (tipicamente 12) pode ser reduzida para 8 o que resulta em taxas de velocidade de transmissão mais lentas mas ao mesmo tempo numa economização do consumo muito significativa;
- O módulo M10 também suporta um modo de consumo inferior (modo *sleep*) que pode ser utilizado na comunicação do controlador SRI com a plataforma central. Mesmo neste estado, o módulo consegue manter a comunicação TCP/IP e receber dados do servidor remoto (só não consegue enviar);

Componentes externos eficientes A eficiência dos componentes externos também pode ter impacto no consumo energético. A escolha desses componentes não pode ser controlada totalmente por nós pois pode variar de cliente para cliente. Contudo, a restrição na alimentação de saída de 3,3V (ver secção 7.2.4) acaba por obrigar à utilização de componentes com tensões baixas e que em princípio não terão um gasto energético considerável.

Conversores de tensão eficientes Esta melhoria já foi abordada na secção 7.2.4. Os conversores de tensão são peças de *hardware* que tipicamente são responsáveis por um desperdício energético considerável e se a sua utilização for mesmo necessária então o tipo e características do conversor devem ser escolhidos com especial atenção.

Capítulo 8

Trabalho Futuro

A natureza multidisciplinar desta tese possibilitou a discussão de várias ideias. Algumas delas não foram implementadas, mas acreditamos no seu potencial. Temos também consciência que há várias características que podem ser melhoradas e que são necessários vários tipos de testes. Este capítulo pretende reunir estas ideias para uma futura implementação.

8.1 Novas funcionalidades

Filtragem de jardins A utilização de um mapa na interface *Web* é útil para ter uma visão global de todos os jardins de uma área, mas pode ser complicado procurar um jardim específico. Uma atualização possível é permitir a pesquisa por morada, nome do jardim ou palavras-chave com redireccionamento automático para a zona do mapa respetiva.

Informação de Histórico Pretendemos adicionar a funcionalidade de histórico no controlador. O controlador pode acordar mais frequentemente sem estabelecer uma ligação remota e apenas fazer a leitura dos sensores. Esta informação será guardada e enviada mais tarde quando for pedida pela plataforma *Web* através de um código específico. Acedendo à gestão do controlador na plataforma central, o utilizador poderá obter o registo de leituras completo.

Vários controladores para um mesmo jardim Poderão existir casos em que um jardim necessite de mais do que um controlador. Sendo assim, não é desejável que todos os controladores estabeleçam comunicação com o sistema central. Nestes casos, e para evitar gastos desnecessários em *hardware* e planos de dados, coloca-se o desafio de criar uma rede privada de comunicação entre os controladores do mesmo jardim. Assim, só um desses controladores comunicará com a plataforma central e será o responsável por transmitir as suas informações e as dos outros controladores.

Programação de uma máquina virtual Após a passagem a produção, prevê-se que continue ativo o desenvolvimento do controlador SRI com vista a melhorar o seu desempenho e possivelmente suportar mais funcionalidades. Este desenvolvimento poderá justificar mudanças no *hardware* do controlador, mas a compatibilidade com versões anteriores da plataforma de gestão é também uma preocupação. A criação de uma máquina virtual será um passo dado neste sentido adicionando uma camada de abstração extra entre as funcionalidades do controlador que são expostas para o sistema central e a forma como essas funcionalidades são implementadas para um controlador específico.

Reprogramação remota do *firmware* do controlador A possibilidade de reprogramar o controlador remotamente é motivada pelos seguintes fatores:

- o controlador será inacessível do exterior pois estará dentro de uma caixa selada e preenchida com resina;
- para melhorar a eficiência energética, os elementos de *hardware* necessários à reprogramação do microcontrolador foram removidos do circuito da placa do controlador;
- será útil atualizar a versão do *software* do controlador corrigindo eventuais erros e adicionando funcionalidades novas;
- a manutenção do controlador poderá ser preferível à sua substituição e consequente destruição sobretudo se disser respeito a uma grande quantidade de controladores;
- será útil adaptar o código do controlador em função dos sensores instalados;

Esta reprogramação é possível se o *bootloader* instalado no microcontrolador estiver preparado para estabelecer uma comunicação remota TCP. Neste caso, a aplicação instalada e executada pelo controlador poderá ser completamente substituída exclusivamente via *software*.

8.2 Melhoramentos de funcionalidades existentes

Consolidação da plataforma central De uma forma geral, as funcionalidades da plataforma central que apresentámos na secção 5 serão revistas no que toca à sua usabilidade e robustez. Por outro lado, as funcionalidades incompletas serão finalizadas como é o caso da gestão de tipos de rega cuja ilustração não foi incluída neste documento.

Mensagens trocadas entre a plataforma central e o controlador SRI As mensagens de reposta do controlador para a plataforma *Web* são enviadas em *ASCII*. Uma atualização

do protocolo de comunicação remota pode passar por uniformizar o formato das mensagens de entrada e saída (ambos sem significado *ASCII*). Esta modificação será especialmente importante se o controlador armazenar dados a serem enviados e onde o tamanho das mensagens pode justificar a passagem para um vetor de *bytes*.

Eficiência energética A preocupação com o consumo energético esteve sempre presente no desenvolvimento deste projeto. As medidas que apontámos na secção 7.3 são importantes para melhorar a eficiência energética do controlador SRI mas este é um trabalho que deverá ter uma continuidade. Por exemplo, uma forma de prolongar a autonomia do controlador SRI é utilizar um pequeno painel solar capaz de recarregar a bateria. Esta será certamente uma solução a ponderar.

8.3 Testes e consumos energéticos

Uma das fases mais importantes deste projeto será a realização de testes e consumos energéticos ao controlador. Os resultados destes testes serão determinantes para perceber se o controlador desenvolvido estará preparado para ser comercializado e, caso não esteja, fazer as alterações necessárias. Os testes serão feitos não só ao protótipo, mas também ao sistema central.

Os testes ao sistema central pretendem apurar os seguintes parâmetros:

- usabilidade da interface *Web*;
- verificação das funcionalidades apontadas durante a fase de requisitos;
- robustez e segurança do sistema.

Os testes ao protótipo prentendem apurar os seguintes parâmetros:

- detalhar os consumos energéticos em diferentes cenários;
- fiabilidade da comunicação via GPRS em diferentes cenários;
- robustez e segurança do protótipo a nível de *software*;
- robustez do circuito desenhado e da placa fabricada;
- capacidade de resistência da caixa do controlador e preparação para um ambiente exposto às condições ambientais, nomeadamente a sua estanquicidade.

Capítulo 9

Conclusão

A possibilidade de pensar e implementar um sistema de rega remoto ofereceu desafios muito interessantes que se estendem desde o desenho de um produto de *hardware* até ao desenvolvimento de uma aplicação *Web* empresarial, passando pela pesquisa de sistemas idênticos e pelo levantamento de requisistos que existem para uma situação da vida real. Adicionalmente, o facto deste projeto ter como finalidade uma aplicação comercial foi uma forte motivação, esperando colher-se esse sucesso num futuro próximo.

Os resultados deste projeto são uma solução de *software* constituída por uma plataforma de gestão que utiliza a tecnologia *Java Enterprise Edition* [Ora13b] e que está instalada no sistema de *nuvem* da *Amazon* [Ama]. Além disso, concebemos um circuito elétrico de raiz, que nos permitirá produzir uma placa de circuito impressa feita à medida. O resultado é um controlador que permite comunicação remota sem fios e ao mesmo tempo altamente eficiente.

A realização deste projeto foi extremamente polivalente. Na área de Engenharia de *software* abraçámos desafios como a programação de dispositivos embebidos com diversas limitações (e.g., pouca memória e capacidade de processamento), utilização de tecnologias empresariais complexas (e.g., Glassfish, JavaEE). Também nas áreas de Redes de Computadores, Sistemas Distribuídos e Segurança Informática através da utilização de GPRS, comunicação TCP por *sockets* na utilização de paradigmas cliente-servidor e *peer-to-peer*. Na área de Eletrónica, através do desenho de um produto final de *hardware* que engloba o desenho e montagem de um circuito eletrónico específico e escolha de materiais adequados a um ambiente de rega. Finalmente, os detalhes relacionados com a gestão de um projeto com clientes reais e a criação de um pedido provisório de patente.

Posso dizer que cada detalhe deste projeto foi feito com imenso prazer e com a vontade sempre presente de aprender mais e utilizar as tecnologias como forma de contribuição de um mundo mais sustentável.

Apêndice A

Casos de uso SRI

A.0.1 Jardins

- Pesquisar jardins
- Gerir jardim
- Adicionar jardim
- Editar jardim
- Remover jardim

Pesquisar jardins

Caso de Uso: Pesquisar jardins.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O sistema devolve a lista de jardins que satisfazem os critérios de pesquisa.
Cenário principal de Sucesso:

1. O operador indica que pretende fazer uma pesquisa de jardins;
2. O sistema devolve uma lista com nomes de filtros para pesquisar jardins;
3. O operador seleciona um filtro;
4. O sistema apresenta um formulário com os campos a preencher para o filtro selecionado;
5. O operador completa as informações para cada filtro;
6. O operador escolhe a opção “Pesquisar”;

7. O sistema devolve a lista dos jardins definidos no sistema contendo os atributos "referência", "nome", "latitude", "longitude" e "estado" com base no filtro selecionado;

8. O operador indica que quer ver mais jardins;

9. O operador repete os passos 7 e 8 enquanto desejar ou até o sistema avisar que não há mais jardins para mostrar.

Extensões:
7A - O sistema mostra a mensagem "Não existem resultados para o filtro selecionado" e retoma-se o cenário principal a partir do passo 2
Observações: O estado do jardim é ativo se pelo menos uma das estações de um dos controladores estiver a regar ou inativo se nenhuma estação estiver a regar.

Gerir jardim

Caso de Uso: Gerir jardim.
Ator principal: Operador.
Pré-condições: Existe um jardim selecionado.
Pós-condições: O sistema devolve as informações associadas ao jardim.
Cenário principal de Sucesso:

1. O operador indica que pretende gerir um jardim;

2. O sistema devolve as informações associadas ao jardim contendo os atributos "nome", "referência", "local", "comentários" e "coordenadas" e uma lista de controladores contendo os atributos "nome", "referência" e "estado";

2A - A lista de controladores está vazia.

Adicionar jardim

Caso de Uso: Adicionar jardim.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O jardim indicado é adicionado ao sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar um jardim;

2. O sistema mostra um mapa;

3. O operador escolhe a zona do mapa onde quer adicionar o novo jardim;

4. O sistema apresenta um formulário com os atributos "nome", "comentários" e "nome do local";

5. O operador fornece a informação e escolhe a opção "Adicionar";

6. O sistema pede para o operador confirmar o pedido;

7. O operador escolhe a opção "Sim";

8. O sistema valida os dados submetidos e mostra uma mensagem a informar que o jardim foi adicionado com sucesso.

Extensões:
2A - O utilizador indica as coordenadas ou a morada do jardim 5A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
7A - O utilizador escolhe a opção "Não" e retorna-se ao passo 4.
8A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 4.

Editar jardim

Caso de Uso: Editar jardim.
Ator principal: Operador.
Pré-condições: Existe um jardim selecionado.
Pós-condições: As informações do jardim selecionado são alteradas e registadas no sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende editar um jardim;

2. O sistema afixa um formulário com as informações atuais do jardim e que podem ser alteradas;

3. O operador faz as alterações desejadas;

4. O operador escolher a opção "Concluído"

5. O sistema pede para o operador confirmar o pedido;

6. O operador escolhe a opção "Sim";

7. O sistema valida os dados submetidos e mostra uma mensagem a informar que o jardim foi atualizado com sucesso.

Extensões:
4A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
6A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
8A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 4.

Remover jardins

Caso de Uso: Remover jardins.
Ator principal: Operador.
Pré-condições: Existe pelo menos um jardim.
Pós-condições: Os jardins selecionados são removidos do sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende remover jardins;
2. O operador seleciona os jardins que pretende remover;
3. O operador escolhe a opção "Concluir";
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção "Sim";
6. O sistema mostra uma mensagem a informar que os jardins selecionados foram removidos com sucesso.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
4A - O sistema mostra uma mensagem de aviso a informar quais os jardins que não é possível remover porque têm controladores com planos de rega em execução e volta-se ao passo 2 do caso de uso.
4B - O sistema mostra uma mensagem de aviso a informar quais os jardins que não é possível remover porque existem ligações ativas de momento com os controladores e volta-se ao passo 2 do caso de uso.
5A - O utilizador escolhe a opção "Não" e volta-se ao passo 2 do caso de uso.
6A - O sistema mostra uma mensagem de erro a informar quais os jardins que não é possível remover porque têm controladores com planos de rega em execução e o caso de uso termina.
6B - O sistema mostra uma mensagem de erro a informar quais os jardins que não é possível remover porque existem ligações ativas de momento com os controladores e o caso de uso termina.

A.0.2 Planos de rega

- Gerir plano de rega
- Adicionar plano de rega
- Editar plano de rega
- Remover plano de rega
- Adicionar evento ao plano de rega
- Remover evento do plano de rega

Gerir plano de rega

Caso de Uso: Gerir plano de rega.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O sistema mostra as informações do plano de rega indicado.
Cenário principal de Sucesso:

1. O operador indica que pretende gerir um plano de rega;
2. O sistema devolve uma lista de planos de rega;
3. O operador seleciona o plano que pretende gerir;
4. O sistema devolve as informações do plano contendo os atributos "nome", "referência", "comentários", "data de entrada em vigor", "tipo de rega" e "estado" e uma lista de eventos;

Extensões:
2A - O sistema mostra a mensagem "Não existem planos de rega criados"e o caso de uso termina.
3A - O operador escolhe a opção "Cancelar" e o caso de uso termina.

Adicionar plano de rega

Caso de Uso: Adicionar plano de rega.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O plano de rega indicado é adicionado ao sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar um plano de rega;
2. O sistema apresenta um formulário com os atributos "nome", "comentários", "data de entrada em vigor" e uma lista de tipos de rega;
3. O operador fornece a informação e escolhe a opção "Adicionar";
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção "Sim";
6. O sistema valida os dados submetidos e mostra uma mensagem a informar que o plano de rega foi adicionado com sucesso.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
5A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 2.

Editar plano de rega

Caso de Uso: Editar plano de rega.
Ator principal: Operador.
Pré-condições: Existe um plano de rega em gestão.
Pós-condições: As informações do plano de rega selecionado são alteradas e registadas no sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende editar um plano de rega;
2. O sistema afixa um formulário com as informações atuais do plano de rega e que podem ser alteradas;
3. O operador faz as alterações desejadas;
4. O operador escolher a opção "Concluído"
5. O sistema pede para o operador confirmar o pedido;
6. O operador escolhe a opção "Sim";
7. O sistema valida os dados submetidos e mostra uma mensagem a informar que o plano de rega foi atualizado com sucesso.

Extensões:
4A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
6A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
8A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 4.

Remover plano de rega

Caso de Uso: Remover plano de rega.
Ator principal: Operador.
Pré-condições: Existe um plano de rega em gestão.
Pós-condições: O plano de rega selecionado é removido do sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende remover um plano de rega;
2. O sistema pede para o operador confirmar o pedido;
3. O operador escolhe a opção "Sim";
4. O sistema mostra uma mensagem a informar que o plano de rega foi removido com sucesso.

Extensões:
3A - O utilizador escolhe a opção "Não" e o caso de uso termina.
4A - O sistema mostra a mensagem de erro "um plano em vigor não pode ser removido" a informar que o plano de rega não foi removido porque está atualmente em vigor e o caso de uso termina.

Adicionar evento ao plano de rega

Caso de Uso: Adicionar evento ao plano.
Ator principal: Operador.
Pré-condições: Existe um plano de rega em gestão.
Pós-condições: É adicionado um evento ao plano de rega em gestão.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar um evento ao plano de rega;
2. O sistema apresenta um formulário contendo os atributos "dia da semana", "hora de início" e "duração";

3. O operador fornece a informação e escolhe a opção “Adicionar”;
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção “Sim”;
6. O sistema valida os dados submetidos e mostra uma mensagem a informar que o evento foi adicionado com sucesso.

Extensões:
3A - O utilizador escolhe a opção “Cancelar” e o caso de uso termina.
5A - O utilizador escolhe a opção “Não” e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro “campos inválidos” a informar quais os campos que contêm valores inválidos e retorna-se ao passo 2.

Remover evento do plano de rega

Caso de Uso: Remover evento do plano de rega.
Ator principal: Operador.
Pré-condições: Existe um plano de rega em gestão.
Pós-condições: O evento indicado é removido do plano de rega em gestão.
Cenário principal de Sucesso:

1. O operador indica que pretende remover um evento;
2. O sistema pede para o operador confirmar o pedido;
3. O operador escolhe a opção “Sim”;
4. O sistema mostra uma mensagem a informar que o evento foi removido com sucesso.

Extensões:
3A - O utilizador escolhe a opção “Não” e o caso de uso termina.

Definir como plano de rega em vigor

Caso de Uso: Definir como plano de rega em vigor.
Ator principal: Operador.
Pré-condições:
Pós-condições: O plano indicado passa a ser o plano ativo para o tipo de rega indicado.
Cenário principal de Sucesso:

1. O operador indica que pretende definir um plano de rega como o plano ativo para o seu tipo de rega associado;

2. O sistema mostra um formulário contendo o campo “data de entrada em vigor”;

3. O operador fornece a informação e escolhe a opção “Definir como plano de rega em vigor”;

4. O sistema mostrar uma mensagem a informar que o plano de rega irá ficar em vigor a partir da data indicada no passo 3.

Extensões:
4A - O sistema mostra a mensagem de erro “a data não é válida” a informar que o plano de rega não pode ficar em vigor.
4B - O sistema mostra a mensagem de erro “já existe outro plano agendado para a mesma data” a informar que o plano de rega não pode ficar em vigor.

A.0.3 Tipos de rega

- Gerir tipo de rega
- Adicionar tipo de rega
- Editar tipo de rega
- Consultar planos de rega associados

Gerir tipo de rega

Caso de Uso: Gerir tipo de rega.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O sistema mostra as informações do tipo de rega indicado.
Cenário principal de Sucesso:

1. O operador indica que pretende gerir um tipo de rega;

2. O sistema devolve uma lista de tipos de rega;

3. O operador seleciona o tipo de rega que pretende gerir;

4. O sistema devolve as informações do tipo de rega contendo os atributos “nome”, “referência”, e “comentários” e uma lista de planos de rega associados;

Extensões:
2A - O sistema mostra a mensagem “Não existem tipos de rega criados”e o caso de uso termina.
3A - O operador escolhe a opção “Cancelar” e o caso de uso termina.

Adicionar tipo de rega

Caso de Uso: Adicionar tipo de rega.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O tipo de rega indicado é adicionado ao sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar um tipo de rega;
2. O sistema apresenta um formulário com os atributos “nome” e “comentários”;
3. O operador fornece a informação e escolhe a opção “Adicionar”;
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção “Sim”;
6. O sistema valida os dados submetidos e mostra uma mensagem a informar que o tipo de rega foi adicionado com sucesso.

Extensões:
3A - O utilizador escolhe a opção “Cancelar” e o caso de uso termina.
5A - O utilizador escolhe a opção “Não” e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro “campos inválidos” a informar quais os campos que contêm valores inválidos e retorna-se ao passo 2.

Editar tipo de rega

Caso de Uso: Editar tipo de rega.
Ator principal: Operador.
Pré-condições: Existe um tipo de rega em gestão.
Pós-condições: As informações do tipo de rega selecionado são alteradas e registadas no sistema.
Cenário principal de Sucesso:

1. O operador seleciona um tipo de rega;

2. O operador indica que pretende editar o tipo de rega;

3. O sistema afixa um formulário com as informações atuais do tipo de rega e que podem ser alteradas;

4. O operador faz as alterações desejadas;

5. O operador escolher a opção “Concluído”

6. O sistema pede para o operador confirmar o pedido;

7. O operador escolhe a opção “Sim”;

8. O sistema valida os dados submetidos e mostra uma mensagem a informar que o tipo de rega foi atualizado com sucesso.

Extensões:
5A - O utilizador escolhe a opção “Cancelar” e o caso de uso termina.
7A - O utilizador escolhe a opção “Não” e retorna-se ao passo 2.
9A - O sistema mostra a mensagem de erro “campos inválidos” a informar quais os campos que contêm valores inválidos e volta ao passo 3.

Consultar planos de rega associados

Caso de Uso: Consultar planos de rega associados.
Ator principal: Operador.
Pré-condições: Existe um tipo de rega em gestão.
Pós-condições: O sistema devolve uma lista com os planos de rega associados ao tipo de rega em gestão.
Cenário principal de Sucesso:

1. O operador indica que pretende consultar os planos de rega associados;

2. O sistema devolve uma lista de planos de rega associados ao tipo de rega em gestão e o caso de uso termina;

Extensões:
2A - O sistema mostra a mensagem “Não existem planos de rega associados”e o caso de uso termina.
3A - O operador escolhe a opção “Cancelar” e o caso de uso termina.

A.0.4 Controladores

- Gerir controlador
- Adicionar controlador
- Editar controlador
- Remover controlador

Gerir controlador

Caso de Uso: Gerir controlador.
Ator principal: Operador.
Pré-condições: Existe um jardim em gestão.
Pós-condições: O sistema mostra as informações do controlador indicado.
Cenário principal de Sucesso:

1. O operador seleciona o controlador que pretende gerir;
2. O sistema devolve as informações do controlador contendo os atributos "nome", "referência", "comentários", "código de barras", "nome do jardim onde está instalado", uma lista de eventos contendo os atributos "referência", "hora de início", "hora de fim", "dia da semana" e "válvula", uma lista de estações de rega contendo os atributos "referência", "tipo de rega"e "estado"e uma lista de sensores contendo os atributos "referência", "nome" e "valor lido" e o caso de uso termina;

Extensões:
2A - O sistema mostra a mensagem "Não existem eventos associados ao controlador indicado".
2B - O sistema mostra a mensagem "Não existem estações de rega associadas ao controlador indicado".
2C - O sistema mostra a mensagem "Não existem sensores associados ao controlador indicado".

Adicionar controlador

Caso de Uso: Adicionar Controlador ao Jardim.
Ator principal: Operador.
Pré-condições: Não existem.
Pós-condições: O controlador indicado é adicionado ao sistema e associado ao jardim atualmente em gestão.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar um controlador;
2. O sistema apresenta um formulário com os atributos "nome", "comentários" e "código de barras";
3. O operador fornece a informação e escolhe a opção "Adicionar";
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção "Sim";
6. O sistema valida os dados submetidos e mostra uma mensagem a informar que o controlador foi adicionado com sucesso.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
5A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 2.

Editar controlador

Caso de Uso: Editar Informações do Controlador do Jardim.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições: As informações do controlador selecionado são alteradas e registadas no sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende editar um controlador;
2. O sistema afixa um formulário com as informações atuais do controlador e que podem ser alteradas;
3. O operador faz as alterações desejadas;
4. O operador escolher a opção "Concluído"
5. O sistema pede para o operador confirmar o pedido;
6. O operador escolhe a opção "Sim";
7. O sistema valida os dados submetidos, mostra uma mensagem a informar que o controlador foi atualizado com sucesso e o caso de uso termina.

Extensões:
4A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
6A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
8A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 4.

Remover controlador

Caso de Uso: Remover Controlador do Jardim.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições: O controlador indicado é removido do sistema, as suas estações são removidas e são também removidas as associações com os sensores respetivos.
Cenário principal de Sucesso:

1. O operador indica que pretende remover o controlador;
2. O sistema pede para o operador confirmar o pedido;
3. O operador escolhe a opção "Sim";
4. O sistema mostra uma mensagem a informar que o controlador foi removido com sucesso e o caso de uso termina.

Extensões:
3A - O utilizador escolhe a opção "Não" e o caso de uso termina.
4A - O sistema mostra a mensagem de erro "só é possível remover o controlador se as suas estações não tiverem um plano de rega ativo" a informar que o controlador não foi removido porque existe pelo menos uma estação de rega do controlador que tem um tipo de rega com um plano ativo.

A.0.5 Estações de rega

- Adicionar estação de rega
- Editar estação de rega
- Remover estação de rega
- Parar rega imediatamente
- Iniciar rega imediatamente
- Desativar a rega temporariamente

Adicionar estação de rega

Caso de Uso: Adicionar estação de rega.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições:
A estação de rega indicada é adicionada ao sistema.
É feita uma associação entre a nova estação e o controlador selecionado.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar uma estação de rega ao controlador;
2. O sistema apresenta um formulário com os atributos "número da porta de instalação da estação no controlador" e uma lista de tipos de rega;
3. O operador fornece a informação e escolhe a opção "Adicionar";
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção "Sim";
6. O sistema valida os dados submetidos, mostra uma mensagem a informar que a estação de rega foi adicionada com sucesso e o caso de uso termina.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
5A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 2.

Editar estação de rega

Caso de Uso: Editar estação de rega.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão com pelo menos uma estação de rega adicionada.
Pós-condições: As informações da estação de rega selecionado são atualizadas e registadas no sistema.
Cenário principal de Sucesso:

1. O operador seleciona uma estação de rega;
2. O operador indica que pretende editar a estação de rega;

3. O sistema afixa um formulário com as informações atuais da estação de rega e que podem ser alteradas;

4. O operador faz as alterações desejadas;

5. O operador escolher a opção "Concluído"

6. O sistema pede para o operador confirmar o pedido;

7. O operador escolhe a opção "Sim";

8. O sistema valida os dados submetidos, mostra uma mensagem a informar que a estação de rega foi atualizada com sucesso e o caso de uso termina.

Extensões:
5A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
7A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
9A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 3.

Remover estações de rega

Caso de Uso: Remover estações de rega.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão com pelo menos uma estação adicionada.
Pós-condições: A estação de rega selecionada é removida do sistema e são atualizados os eventos de rega do controlador.
Cenário principal de Sucesso:

1. O operador indica que pretende remover estações de rega;

2. O operador seleciona as estações de rega que pretende remover;

3. O operador escolhe a opção "Concluir";

4. O sistema pede para o operador confirmar o pedido;

5. O operador escolhe a opção "Sim";

6. O sistema mostra uma mensagem a informar que a estação de rega foi removida com sucesso e o caso de uso termina.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
4A - O sistema mostra uma mensagem de aviso a informar quais as estações de rega que não é possível remover porque só é possível remover uma estação de rega se o seu tipo não tiver um plano de rega ativo e volta-se ao passo 2 do caso de uso.
5A - O utilizador escolhe a opção "Não" e volta-se ao passo 2 do caso de uso
6A - O sistema mostra a mensagem de erro "só é possível remover uma estação de rega se o seu tipo não tiver um plano de rega ativo" a informar que a estação de rega não foi removida porque existe um plano que está em vigor no tipo de rega dessa estação e o caso de uso termina.

Parar a rega imediatamente

Caso de Uso: Parar a rega imediatamente.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições: A rega é interrompida nas estações de rega selecionadas.
Cenário principal de Sucesso:

1. O operador seleciona uma ou mais estações de rega;
2. O operador indica que pretende interromper a rega nas estações de rega selecionadas;
3. O sistema pede para o operador confirmar o pedido;
4. O operador escolhe a opção "Sim";
5. O sistema mostra uma mensagem a informar que a rega foi interrompida nas estações de rega selecionadas e o caso de uso termina;

Extensões:
4A - O utilizador escolhe a opção "Não" e o caso de uso termina.
5A - O sistema mostra a mensagem de erro "só pode parar a rega se a estação estiver ativa" e o caso de uso termina.

Iniciar a rega imediatamente

Caso de Uso: Iniciar a rega imediatamente.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.

Pós-condições: A rega é iniciada nas estações de rega selecionadas.
Cenário principal de Sucesso:

1. O operador seleciona uma ou mais estações de rega;
2. O operador indica que pretende iniciar a rega nas estações de rega selecionadas;
3. O sistema pede para o operador confirmar o pedido;
4. O operador escolhe a opção "Sim";
5. O sistema mostra uma mensagem a informar que a rega foi iniciada nas estações de rega selecionadas e o caso de uso termina;

Extensões:
4A - O utilizador escolhe a opção "Não" e o caso de uso termina.
5A - O sistema mostra a mensagem de erro "só pode iniciar a rega se a estação estiver desativa" e o caso de uso termina.

Desativar a rega temporariamente

Caso de Uso: Desativar a rega temporariamente.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições: A rega é desativada nas estações selecionadas pelo período de tempo definido pelo operador.
Cenário principal de Sucesso:

1. O operador seleciona uma ou mais estações de rega;
2. O operador indica que pretende desativar a rega temporariamente nas estações de rega selecionadas;
3. O sistema pede para o operador confirmar o pedido;
4. O operador escolhe a opção "Sim";
5. O sistema mostra uma mensagem a informar que a rega estará desativada nas estações de rega selecionadas até à data indicada pelo operador e o caso de uso termina;

Extensões:
4A - O utilizador escolhe a opção "Não" e o caso de uso termina.
5A - O sistema mostra a mensagem de erro "as estações de rega selecionadas já se encontram desativadas até à data definida" e o caso de uso termina.

A.0.6 Sensores

- Adicionar sensor
- Editar sensor
- Remover sensor
- Consultar registos do sensor

Adicionar sensor

Caso de Uso: Adicionar sensor.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão.
Pós-condições: O sensor indicado é adicionado ao sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende adicionar um sensor ao controlador;
2. O sistema apresenta um formulário com os atributos "nome", "código", "comentários" e "especificações";
3. O operador fornece a informação e escolhe a opção "Adicionar";
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção "Sim";
6. O sistema valida os dados submetidos, mostra uma mensagem a informar que o sensor foi adicionado com sucesso e o caso de uso termina.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
5A - O utilizador escolhe a opção "Não" e retorna-se ao passo 2.
6A - O sistema mostra a mensagem de erro "campos inválidos" a informar quais os campos que contêm valores inválidos e volta ao passo 2.

Editar sensor

Caso de Uso: Editar sensor.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão com pelo menos um sensor adicionado.
Pós-condições: As informações do sensor selecionado são atualizadas e registadas no sistema.
Cenário principal de Sucesso:

1. O operador seleciona um sensor;
2. O operador indica que pretende editar o sensor;
3. O sistema afixa um formulário com as informações atuais do sensor e que podem ser alteradas;
4. O operador faz as alterações desejadas;
5. O operador escolher a opção “Concluído”
6. O sistema pede para o operador confirmar o pedido;
7. O operador escolhe a opção “Sim”;
8. O sistema valida os dados submetidos, mostra uma mensagem a informar que o sensor foi atualizado com sucesso e o caso de uso termina.

Extensões:
5A - O utilizador escolhe a opção “Cancelar” e o caso de uso termina.
7A - O utilizador escolhe a opção “Não” e retorna-se ao passo 2.
9A - O sistema mostra a mensagem de erro “campos inválidos” a informar quais os campos que contêm valores inválidos e volta ao passo 3.

Remover sensores

Caso de Uso: Remover sensores.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão com pelo menos um sensor adicionado.
Pós-condições: Os sensores selecionados são removidos do sistema.
Cenário principal de Sucesso:

1. O operador indica que pretende remover sensores;
2. O operador seleciona os sensores de rega que pretende remover;
3. O operador escolhe a opção “Concluir”;
4. O sistema pede para o operador confirmar o pedido;
5. O operador escolhe a opção “Sim”;
6. O sistema mostra uma mensagem a informar que os sensores foram removidos com sucesso e o caso de uso termina.

Extensões:
3A - O utilizador escolhe a opção "Cancelar" e o caso de uso termina.
5A - O utilizador escolhe a opção "Não" e volta-se ao passo 2 do caso de uso

Consultar registos do sensor

Caso de Uso: Consultar registos do sensor.
Ator principal: Operador.
Pré-condições: Existe um controlador em gestão com pelo menos um sensor adicionado.
Pós-condições: O sistema apresenta uma lista de registos com as leituras dos sensores selecionados.
Cenário principal de Sucesso:

1. O operador seleciona um ou mais sensores;
2. O operador indica que pretende consultar os registos dos sensores;
3. O sistema devolve uma lista de registos com as leituras dos sensores selecionados e o caso de uso termina.

Extensões:
3A - Não existem registos para os sensores selecionados e o caso de uso termina.

Apêndice B

Modelo Entidade-Relação da base de dados SRIDB

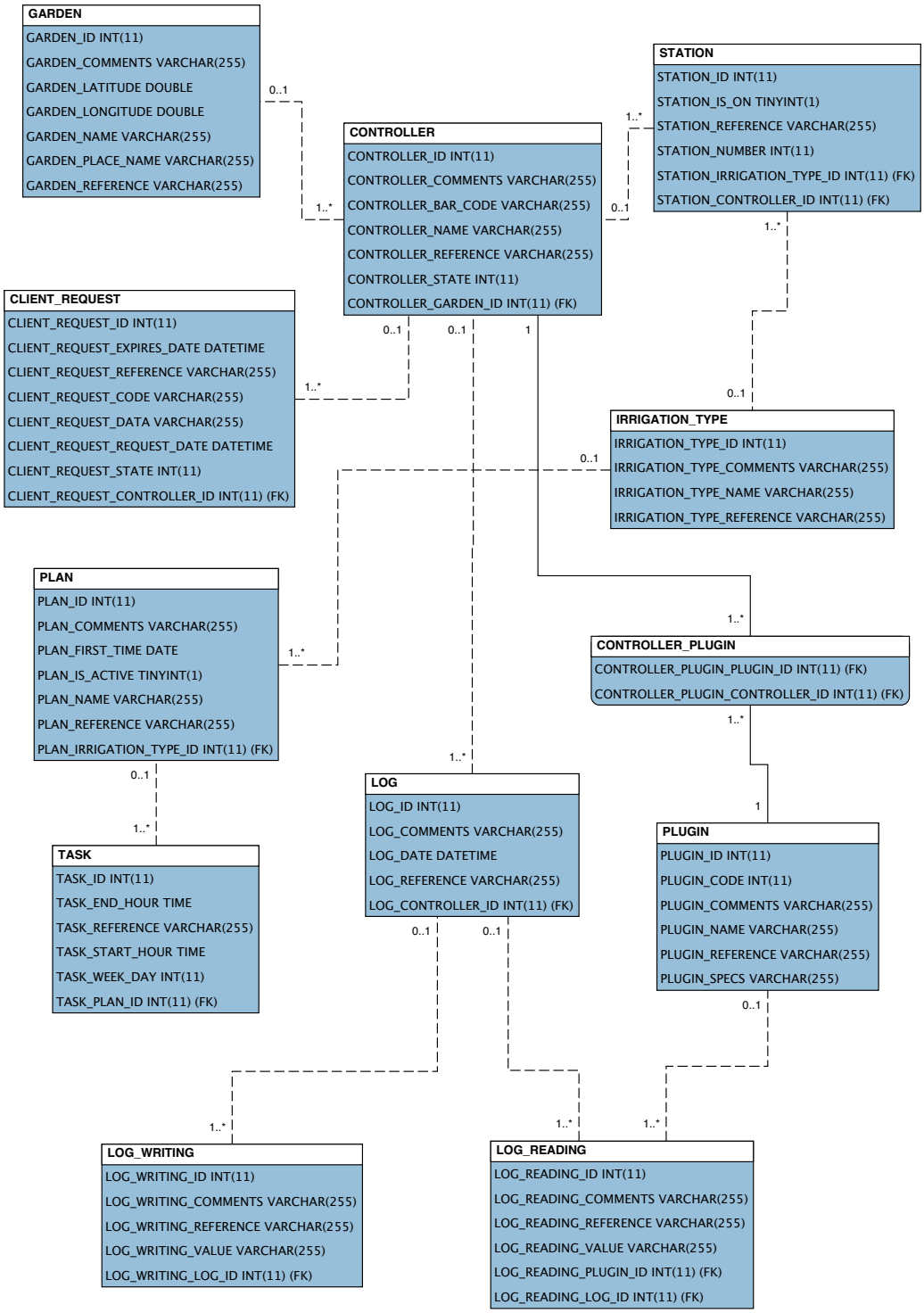


Figura B.1: Modelo Entidade-Relação da base de dados SRIDB

Apêndice C

Protocolo de Comunicação Remota

C.1 Códigos

Pedir o identificador do controlador

código do pedido	10
código da resposta	10
descrição	Pede o identificador do controlador
formato do pedido	<id_servidor><10><255>
formato da resposta	1:2#10:<id_controlador>#

Pedir a temperatura do ar

código do pedido	11
código da resposta	11
descrição	Pede a temperatura do ar atual em graus Celsius.
formato do pedido	<id_servidor><11><255>
formato da resposta	1:2#10:<id_controlador>#11:<temperatura>#

Pedir a humidade do ar

código do pedido	12
código da resposta	12
descrição	Pede a humidade relativa do ar atual (percentagem).
formato do pedido	<id_servidor><12><255>
formato da resposta	1:2#10:<id_controlador>#12:<humidade>#

Pedir a humidade do solo

código do pedido	13
código da resposta	13
descrição	Devolve um valor entre 0 e 950 que representa a quantidade de humidade do solo.
formato do pedido	<id_servidor><13><255>
formato da resposta	1:2#10:<id_controlador>#13:<humidade_do_solo>#

Pedir o nível das baterias do controlador

código do pedido	14
código da resposta	14
descrição	Devolve o valor das baterias do controlador em Volts.
formato do pedido	<id_servidor><14><255>
formato da resposta	1:2#10:<id_controlador>#14:<nivel_bateria>#

Pedir o estado de todas as estações do controlador

código do pedido	15
código da resposta	15
descrição	Devolve o estado atual de cada uma das estações do controlador.
formato do pedido	<id_servidor><15><255>
formato da resposta	1:2#10:<id_controlador>#15:<array_de_inteiros>#

Pedir todas as informações

código do pedido	99
código da resposta	11, 12, 13, 14, 15
descrição	Devolve a informação associada a todos os dados do controlador.
formato do pedido	<id_servidor><99><255>
formato da resposta	1:2#10:<id_controlador>#11:<temperatura>#12:<humidade># 13:<humidade_do_solo>#14:<nivel_bateria># 15:<array_de_inteiros>#

Ativar/Desativar uma válvula

código do pedido	101
código da resposta	ver estado do pedido
descrição	Altera o estado de uma válvula. Para ligar a válvula é enviado o valor 1 e para desligar o valor 0.
formato do pedido	<id_servidor><101><2><numero_valvula><valor_valvula>
formato da resposta	1:2#10:<id_controlador>#

Adormecer o controlador

código do pedido	200
código da resposta	ver estado do pedido
descrição	Não há mais pedidos para fazer e o controlador pode adormecer.
formato do pedido	<id_servidor><200><0>
formato da resposta	1:2#10:<id_controlador>#

Estado do Pedido

Código do pedido	**Sigla**	**Descrição**
0	NOTOK	não foi possível executar o pedido porque ocorreu um erro
1	RESEND	foi excedido o tempo de espera por um pedido do servidor e o controlador pede que seja enviado de novo o mesmo pedido
2	OK	o pedido foi executado com sucesso

C.2 Estrutura de um evento do plano de rega

A estrutura que apresentamos de seguida é utilizada quando é enviado um novo plano de rega para o controlador. Esta estrutura é utilizada também para armazenar o plano no controlador.

Nome	Tamanho máx. (bits)	Valores válidos e seu significado	Total desperdiçado (bits)
Válvula	$2^5 = 32$	[1,31]	1
Dia	$2^3 = 8$	[1,7] em que 1=Domingo, 2=Segunda, 3=Terça, 4=Quarta, 5=Quinta, 6=Sexta e 7=Sábado	1
Hora	$2^5 = 32$	[1,24] em que 1=1h, 2=2h, 3=3h, (...), 22=22h, 23=23h, 24=0h	9
Duração (horas)	$2^3 = 8$	[1,7] em que 1=1h, 2=2h, 3=3h, 4=4h, 5=5h, 6=6h, 7=0h	1
Minutos	$2^4 = 16$	[1,12] em que 1=5min, 2=10min, 3=15min, 4=20min, 5=25min, 6=30min, 7=35min, 8=40min, 9=45min, 10=50min, 11=55min, 12=0min	5

C.3 Diagrama de Comunicação

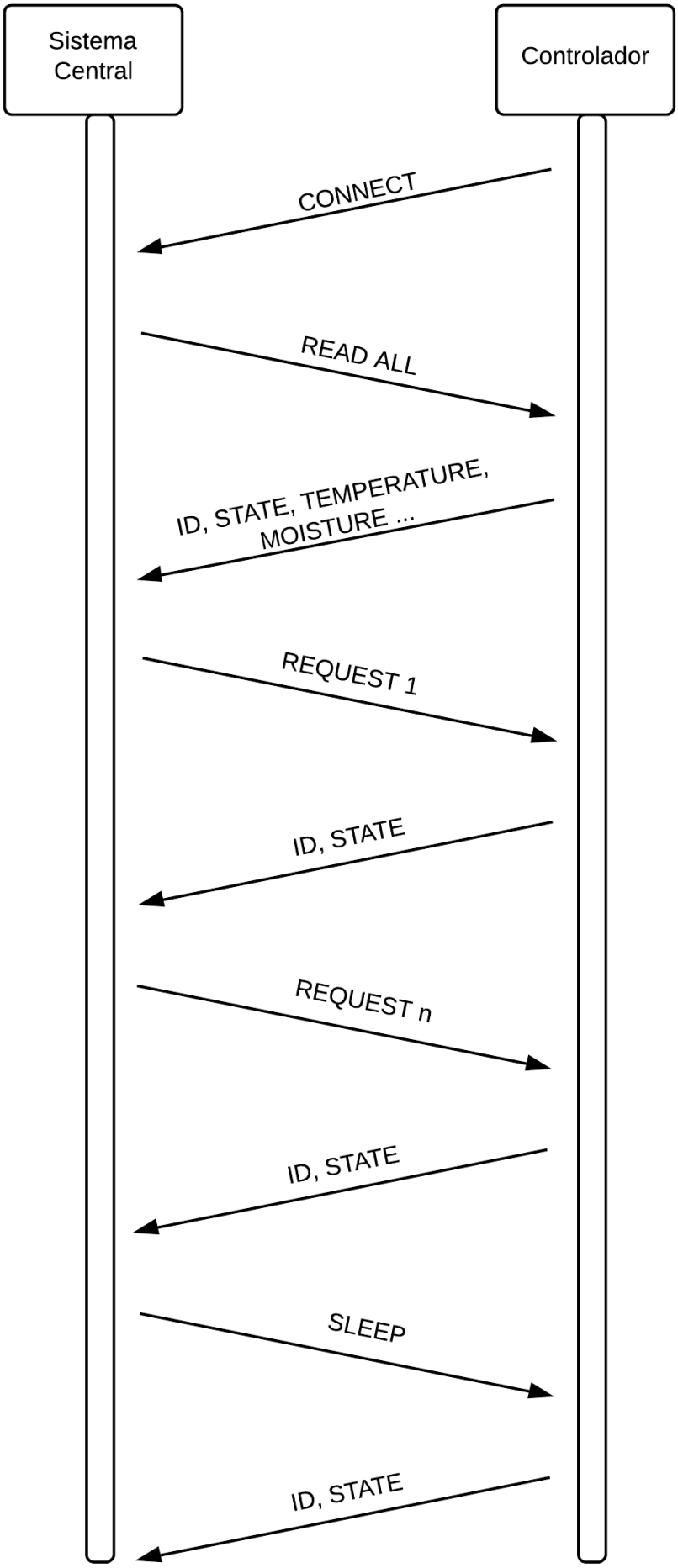


Figura C.1: Diagrama do Protocolo de Comunicação

Apêndice D

Controlo de uma válvula solenóide com auxílio de um transístor

Uma válvula solenóide é um dispositivo que permite controlar a passagem de um líquido como é o caso da rega em que a válvula é utilizada para iniciar e parar a passagem da água. O funcionamento da válvula solenóide assenta na aplicação do eletromagnetismo. A válvula é composta por uma câmara que contém um objeto de metal no interior que se pode deslocar (pistão). Esta câmara é envolvida por uma bobina que devido à aplicação de corrente, gera um campo magnético que por sua vez provoca a deslocação do pistão dentro da câmara. A esta câmara está ligada uma conduta que permite a passagem de um líquido. Se o objeto metálico estiver na posição inferior o líquido não pode fluir criando-se pressão apenas num dos lados da conduta. Essa é aliás a posição por omissão devido à existência de uma mola que obriga o objeto a manter-se na posição inferior. Por outro lado, se o objeto se deslocar, o líquido pode passar livremente.

Percebe-se assim como a válvula pode ser usada para ativar e desativar a rega. Se a válvula for alimentada com uma certa tensão, à qual corresponde a passagem de corrente na bobina, o objeto desloca-se e o líquido passa, caso contrário, o líquido não flui. No entanto, fica por explicar quem controla a alimentação da válvula. A utilização de um dos pinos do microcontrolador seria aparentemente suficiente para fazer esse controlo pois cada pino pode ser programado como ligado ou desligado (passagem ou não passagem de corrente, respetivamente). No entanto, a tensão de funcionamento do microcontrolador é inferior a 9V (5V no caso do protótipo e 3,3V no caso do controlador SRI), enquanto que a válvula solenóide precisa de 9V. Assim, a válvula tem que ser alimentada a partir de uma fonte de tensão dedicada e independente da tensão de operação do microcontrolador.

Para permitir o controlo da válvula utilizando o controlador e ainda assim utilizar alimentações distintas foi projetado um circuito adicional tendo por base um transístor bipolar. Um transístor é um componente eletrónico que pode ser utilizado como uma resistência variável controlada em tensão.

Um transístor é formado por três partes semicondutoras. As duas partes exteriores são do mesmo semicondutor mas a parte do meio é um semicondutor diferente. Semi-

condutores são materiais feitos a partir de metalóides (e.g., Silício) que são dopados com outros elementos. Um metalóide é mau condutor de eletricidade mas quando dopado com certos elementos (e.g., Fósforo) ganha maior condutividade. Daí o nome semicondutor, que se aplica por serem materiais com um nível de condutividade inferior à dos metais (e.g., Cobre) mas superior aos não-metais (e.g., Carbono). Os dois semicondutores utilizados na constituição de um transístor são dopados com elementos diferentes de tal forma que estando ligados entre si numa configuração SC1a-SC2-SC1b e aplicando tensão entre SC1a e SC1b não há passagem de corrente elétrica. Isso acontece pois a falta de eletrões num dos semicondutores é preenchida pelo excesso de eletrões do outro semicondutor, deixando de existir eletrões livres que formem uma corrente elétrica. Entre os dois semicondutores cria-se uma barreira chamada de zona de depleção. Contudo existe uma exceção a esta condição: aplicando uma tensão (a partir de um certo valor) ao semicondutor do meio independente da tensão aplicada a SC1a e SC1b, reúnem-se novamente condições para a corrente fluir entre os dois semicondutores das pontas.A figura D.1 ilustra a composição de um transístor.

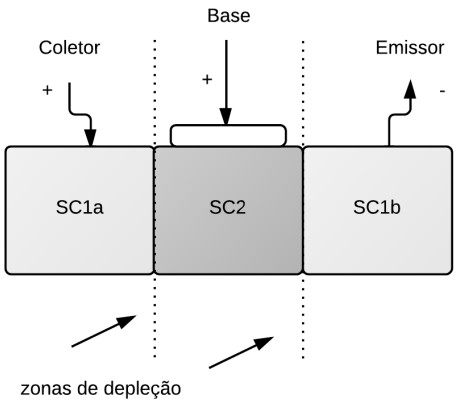


Figura D.1: Composição de um transístor

Associados às três partes semicondutoras de um transístor estão três pinos, dois com função de entrada de tensão e um com função de massa. O transístor pode receber duas alimentações diferentes. A estes três pinos dá-se a designação de emissor, coletor e base: o emissor é a entrada da tensão que serve para alimentar outro circuito, o coletor é a ligação à massa e a base é a entrada da tensão que permite controlar a passagem de corrente entre o emissor e o coletor. Introduzindo os conceitos referidos no parágrafo anterior, o emissor e o coletor são ligações aos semicondutores das pontas SC1a e SC1b e a base é a ligação ao semicondutor do meio SC2. Como a tensão aplicada na base pode controlar uma tensão diferente (tipicamente superior) aos terminais do emissor e do coletor diz-se que os transístores são amplificadores de sinal. Adicionalmente, os transístores

podem comportar-se também como interruptores. É o caso específico em que não há tensão aplicada na base e não flui corrente entre o emissor e o coletor (estado 0 ou OFF) ou em que há saturação do semicondutor do meio e a corrente passa entre o emissor e o coletor com uma resistência perto de zero (estado 1 ou ON).

Importa agora contextualizarmos a utilização do transístor num circuito divisor de tensão. Um circuito deste tipo tem a propriedade da tensão de saída ser igual ou inferior à tensão de entrada em função da resistência dos elementos que o constituem. A figura D.2 ilustra o diagrama deste circuito e na tabela D.1 são exemplificados quatros cenários em que a tensão de saída é alterada com base na variação da resistência R2.

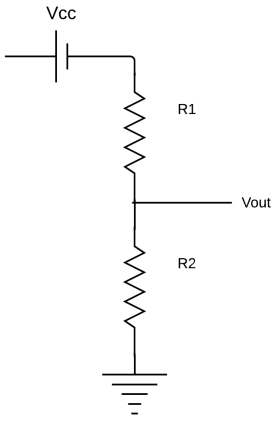


Figura D.2: Circuito necessário para uma válvula solenóide

Vcc	**R1**	**R2**	**Vout**
9V	100Ω	1,5KΩ	6.0000V
9V	100Ω	200Ω	8,4375V
9V	100Ω	0.1Ω	~0,0089V (~0V)
9V	100Ω	10MΩ	~8,9999V (~9V)

Tabela D.1: Variação da tensão de saída em fução de R2

Resumindo, pode dizer-se que quanto mais elevado for o valor de R2 maior é a tensão de saída (até ao nível máximo que se aproxima muito do valor da tensão de entrada) e quanto menos elevado for o valor de R2 menor é a tensão de saída (até ao nível mínimo que se aproxima do valor zero). Este circuito tem diversas utilidades mas tem uma limitação: o valor da tensão de saída depende dos valores relativos das resistências R1 e R2. O ideal seria ter uma forma de controlar o valor de uma das resistências e consequentemente controlar a tensão de saída. É exatamente esse o papel do transístor quando usado num circuito divisor de tensão. A figura D.3 ilustra o diagrama do circuito e na

tabela D.2 são exemplificados três cenários em que a tensão de saída é alterada com base na variação da resistência do transístor (que substitui R2).

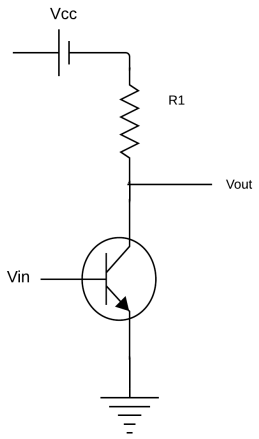


Figura D.3: Circuito necessário para uma válvula solenóide

Vin	**Vcc**	**R1**	**Transístor (R2)**	**Vout**
0	9,0V	100Ω	$+\infty\Omega$	~9V
$0 <$ Vin $<$ 3,3V	9,0V	100Ω	$0 <$ R2 $< +\infty$	$0 <$ Vout $<$ 9,0V
~3,3V	9,0V	100Ω	~0Ω	~0V

Tabela D.2: Variação da tensão de saída em função da variação da resistência do transístor

A tabela D.3 relaciona a variação da tensão de entrada (tensão entre a base e o emissor do transístor) com a tensão de saída recebida pela válvula solenóide e que é alterada pelo controlador. Na configuração descrita a válvula está ligada (ON) quando o pino do controlador não tem tensão (OFF) e vice-versa. Esta condição não é a mais desejável pois obriga a que o pino do controlador tenha uma tensão não nula para que a válvula esteja desligada. Uma das melhorias que pretendemos introduzir no futuro é alterar este cenário e assim aumentar o rendimento energético do controlador.

Finalmente, importa referir que é conveniente colocar um díodo entre o transístor e a válvula evitando picos de descargas de corrente no sentido válvula-controlador, ou seja, evitar que o controlador receba uma tensão superior a 5V proveniente da fonte de alimentação dedicada da válvula de 9V. A figura D.4 ilustra o circuito relacionado com a válvula solenóide, o transístor e o díodo.

Pino do microcontrolador	**Vcc**	**Transístor** (R2)	**Vout**
OFF (~0V)	9,0V	$+\infty\Omega$	(~9V) ON Há tensão suficiente para se criar o campo magnético e o pistão levanta deixando a água circular.
ON (~3,3V)	9,0V	0Ω	(~0V) OFF Sem tensão aplicada, o campo magnético desaparece e o pistão volta à posição inferior, logo não há circulação da água.

Tabela D.3: Relação da tensão de entrada do controlador com a tensão de saída recebida pela válvula solenóide

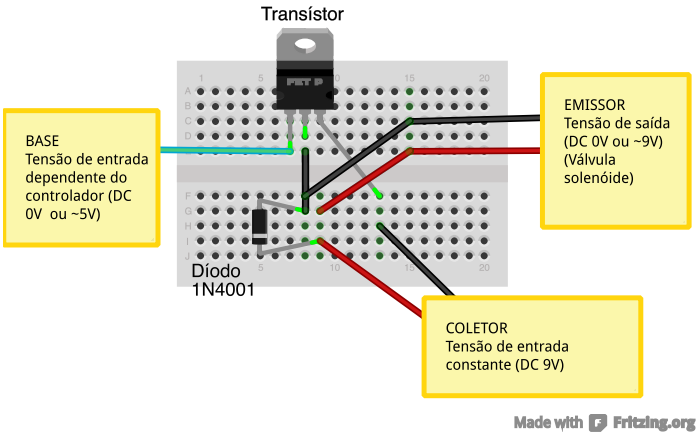


Figura D.4: Circuito de controlo da válvula solenóide

Apêndice E

Configuração do microcontrolador AVR ATMEGA644

A configuração do *bootloader* e dos *fuses* é um passo importante na preparação do microcontrolador pois há diversos parâmetros que têm impacto no desempenho e rendimento energético deste. A plataforma Arduino facilita o processo de configuração do microcontrolador fornecendo já várias configurações para os microcontroladores mais comuns. No caso do microcontrolador ATMEGA644 são necessárias as seguintes modificações:

1. configuração do ficheiro "boards.txt" que contém as definições do protocolo de comunicação, do *bootloader* e do microcontrolador;

2. configuração do ficheiro "pins_arduino.h" que mapeia os pinos do controlador para uma biblioteca de *software*;

3. configuração do ficheiro "makefile" que gera o código compilado do *bootloader*.

A configuração que utilizamos visa a poupança energética máxima sem que o desempenho seja demasiado sacrificado. De seguida apresentamos a configuração do ficheiro "boards.txt".

```
#A. Referencia da configuracao
atmega644f.name=SRI com ATMEGA644

#B. Comunicacao
atmega644f.upload.protocol=stk500
atmega644f.upload.maximum_size=63488
atmega644f.upload.speed=19200

#C. Bootloader
atmega644f.bootloader.low_fuses=0xE2
atmega644f.bootloader.high_fuses=0xDC
atmega644f.bootloader.extended_fuses=0xFF
atmega644f.bootloader.path=atmega
```

```
atmega644f.bootloader.file=ATmegaBOOT_168_atmega644.hex
atmega644f.bootloader.unlock_bits=0x3F
atmega644f.bootloader.lock_bits=0x2F

#D. Compilacao
atmega644f.build.mcu=atmega644
atmega644f.build.f_cpu=8000000L
atmega644f.build.core=arduino
atmega644f.build.variant=sri
```

Como se pode ver neste ficheiro, existem quatro secções principais. As tabelas E.1, E.2 e E.3 detalham o significado de cada uma das propriedades definidas e apresentam uma justificação para a escolha desses valores.

Nome	**Valor**	**Significado**	**Justificação**
A. Referência da configuração			
nome	SRI com ATMEGA644	O ficheiro "boards.txt"pode ter várias configurações que contêmm parâmetros específicos para cada microcontrolador. Estas configurações são depois acessíveis a partir da interface do editor Arduino.	Nome que referencia a utilização do microcontrolador ATMEGA644.
B. Comunicação			
protocolo	stk500	protocolo de comunicação	é o protocolo mais comum e não tem nenhum impacto na eficiência do controlador.
tamanho máximo	63488	n/a	n/a
velocidade	19200	n/a	n/a

Tabela E.1: Propriedades do protocolo de comunicação utilizado para carregar os dados no microcontrolador

C. Bootloader			
Nome	**Valor**	**Significado**	**Justificação**
low fuses	0xE2	é utilizado o oscilador interno e uma velocidade de relógio de 8MHz	Esta frequência é o valor mais baixo que permite maior rendimento energético sem sacrificar demasiado o desempenho. Com esta frequência o microcontrolador pode funcionar com apenas 3,3V e não necessita de um oscilador externo que consome mais energia.
high fuses	0xDC	ativa o protocolo SPI, define o tamanho do *bootloader* (1024 *words*), define o endereço de início do *bootloader* (0x7C00), ativa arranque automático quando é feito *reset*, desativa a funcionalidade JTAG	O tamanho do *bootloader* varia com as suas características e neste projeto queremos que ocupe o mínimo possível deixando mais espaço para a aplicação. O arranque automático e protocolo SPI são configurações necessárias ao contrário de JTAG que não iremos utilizar.
extended fuses	0xFF	desativa a funcionalidade de *brown-out detection*	Esta funcionalidade permite que o microcontrolador seja desativado se atingir limites extremos de tensão (e.g., tensão de alimentação é muito baixa) mas tem um custo no consumo energético. Para o controlador SRI, assumimos que a alimentação será fornecida adequadamente aumentando o rendimento energético.
caminho do ficheiro	atmega	nome da pasta onde está o ficheiro de *bootloader*	n/a
nome do ficheiro	*	nome do ficheiro que contém o código compilado do *bootloader*	n/a
unlock bits	0x3F	.	.
lock bits	0x2F	.	.

Tabela E.2: Propriedades do *bootloader*

D. Compilação			
Nome	**Valor**	**Significado**	**Justificação**
identificador do micro-controlador	atmega644	n/a	Este valor varia em função do microcontrolador utilizado e consoante o identificador indicado será gerado um *bootloader* com características diferentes.
frequência	8000000L	velocidade de relógio	este valor vai gerar um *bootloader* com características diferentes e deve estar de acordo com a propriedade *low fuses*.
bibliotecas	arduino	utiliza as bibliotecas de funcionamento do Arduino	O controlador SRI utiliza as bibliotecas do Arduino responsáveis pelas funcionalidades básicas do controlador (e.g., funções de entrada/saída, definição de tipos de dados, etc.)
variantes	sri	utiliza um ficheiro 'pins_arduino.h" específico que permite mapear os pinos do microcontrolador para uma biblioteca de *software*	A configuração que aqui usamos é baseada no projeto Sanguino [Hoe08]

Tabela E.3: Propriedades do processo de compilação

Apêndice F

Controlador SRI - Esquema do circuito do controlador SRI

De seguida, apresentamos o esquema do circuito do controlador SRI. Para fazer este desenho utilizámos uma das ferramentas mais comuns para o desenho profissional de circuitos, o EAGLE [Cad].

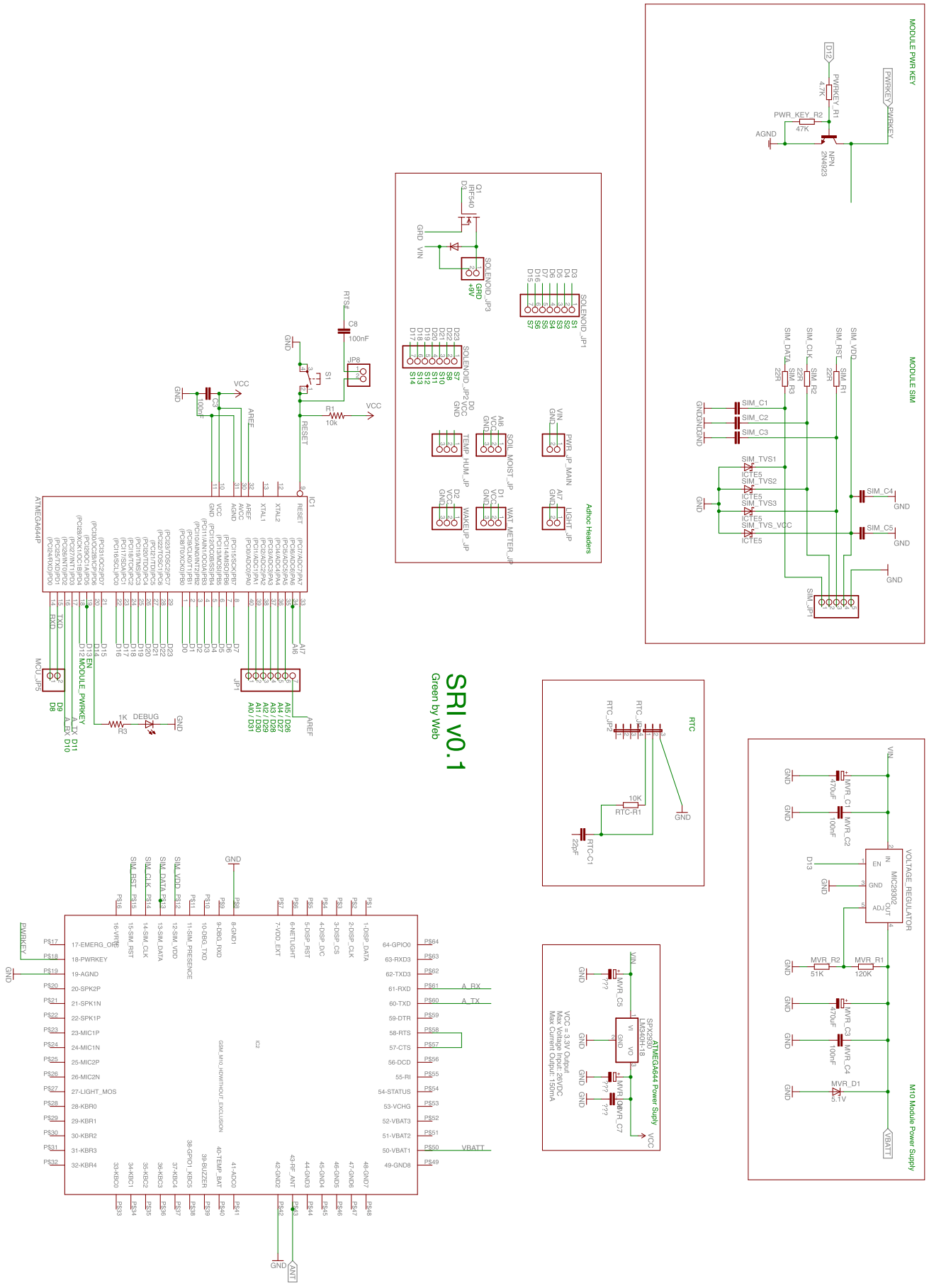


Figura F.1: Esquema do circuito do controlador SRI

Bibliografia

[Ama] Amazon. Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/pt/ec2/. Acedido em: 2013-08-02.

[App13] Apple. Calendário da apple. http://www.apple.com/support/ical/, Janeiro 2013.

[Ard] Arduino. Biblioteca spi. http://arduino.cc/en/Reference/SPI. Acedido em: 2013-07-29.

[Ard12] Arduino. Arduino board uno. http://arduino.cc/en/Main/ArduinoBoardUno, Dezembro 2012.

[Ard13] Arduino. Arduino. http://arduino.cc, Junho 2013.

[ATM] ATMEL. Microcontrolador avr atmega328p. http://www.atmel.com/devices/ATMEGA328P.aspx. Acedido em: 2013-05-14.

[Cad] CadSoft. Cadsoft eagle pcb design software. http://www.cadsoftusa.com/. Acedido em: 2013-09-02.

[Cal13] Calsense. Controladores calsense. http://www.calsense.com/controllers/, Janeiro 2013.

[Cat10] Cathedrow. Narcoleptic. http://code.google.com/p/narcoleptic/, Julho 2010.

[CJ02] David Chappel and Tyler Jewell. *Java Web Services*. O'Reilly, mar 2002.

[Cyb13] CyberRain. Sistemas cyber rain profissionais. http://www.cyberrain.com/professional-irrigation-solutions.html, Janeiro 2013.

[Dum12] Adriane Dummer. Consumos de água para rega em jardins municipais do concelho de lisboa: proposta para uma gestão mais eficiente. http://repositorio.ul.pt/bitstream/10451/7499/1/ulfc099197_tm_adriane_dummer.pdf/, 2012. Acedido em: 2012-10-24.

[Eco13] Ecocasa. Utilização eficiente da água nos espaços verdes. `http://ecocasa.pt/agua_content.php?id=45`, Janeiro 2013. Acedido em: 2012-10-24.

[Eng13a] EngBlaze. Avr and arduino sleep mode basics. http://www.engblaze.com/hush-little-microprocessor-avr-and-arduino-sleep-mode-basics/, Janeiro 2013.

[Eng13b] EngBlaze. Tutorial on arduino interrupts. http://www.engblaze.com/we-interrupt-this-program-to-bring-you-a-tutorial-on-arduino-interrupts/, Janeiro 2013.

[EPO13] EPO. European patent office. `http://www.epo.org/`, Junho 2013. Acedido em: 2013-05-14.

[ETW13] ETWater. Produtos et water. http://www.etwater.com/public/support-product-literature.html, Janeiro 2013.

[Fow06] Martin Fowler. Graphical user interface architectures - model view controller. `http://martinfowler.com/eaaDev/uiArchs.html#ModelViewController`, Julho 2006. Acedido em: 2013-08-01.

[Fri13] Fritzing. Fritzing beta software. `http://fritzing.org`, Janeiro 2013.

[Goo13a] Google. Calendário da google. http://calendar.google.com, Janeiro 2013.

[Goo13b] Google. Google maps api. https://developers.google.com/maps/, Janeiro 2013.

[Hoe08] Zach Hoeken. Controlador sanguino. `http://sanguino.cc/hardware`, Julho 2008. Acedido em: 2013-04-14.

[Hun13] Hunter. Sistemas hunter - imms. http://www.hunterindustries.com/irrigation-product/central-control/imms, Janeiro 2013.

[Hyd13a] Hydropoint. Sistemas hydropoint - weathertrak. http://www.hydropoint.com/wordpress/products/, Janeiro 2013.

[Hyd13b] Hydrosaver. Sistemas hydrosaver. http://www.hydrosaver.net/Items.aspx?catId=c62, 2013.

[IET] IETF. icalendar. http://www.ietf.org/rfc/rfc2445.txt.

[Irr13] Irrisoft. Sistemas irrisoft. http://www.irrisoft.net/index.htm, Janeiro 2013.

[Jef10] Mark Jeffrey. Jca sockets. `https://code.google.com/p/jca-sockets/`, Julho 2010. Acedido em: 2013-06-14.

[Lar01] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall PTR, 2 edition, 2001.

[Mic] Microchip. In-circuit serial programming guide. `http://ww1.microchip.com/downloads/en/devicedoc/30277d.pdf`. Acedido em: 2013-07-29.

[Mor10] Donal Morrisey. Sleeping arduino. http://donalmorrissey.blogspot.pt/2010/04/putting-arduino-diecimila-to-sleep-part.html, Abril 2010.

[Ora] Oracle. Glassfish - open source application server. `https://glassfish.java.net`. Acedido em: 2013-08-01.

[Ora13a] Oracle. Java ee 5 tutorial. http://docs.oracle.com/javaee/5/tutorial/doc/, Janeiro 2013.

[Ora13b] Oracle. Java ee 6 tutorial. `http://docs.oracle.com/javaee/6/tutorial/doc/`, Janeiro 2013.

[oti12] U.S. Department of the interior. Reclamation - managing water in the west. Technical report, U.S. Department of the interior, Denver, Colorado, Julho 2012.

[Pfi11] Cuno Pfister. *Getting Started with the Internet of Things*. O'Reilly Media, Inc., 1 edition, 2011.

[Que13] Quectel. Quectel - wireless module expert. `http://www.quectel.com/`, Abril 2013. Acedido em: 2013-04-14.

[Rai13a] Rainbird. Controladores rainbird. http://www.rainbird.com/landscape/products/controllers/index Janeiro 2013.

[Rai13b] Rainbird. Rainbird wp series. http://www.rainbird.eu/5-6558-Fiche-produit.php?id_produits=44, Janeiro 2013.

[Rai13c] Rainbird. Sistemas rainbird - iq central control. http://www.rainbird.com/ESPLXseries/products/central/IQv2.htm, Janeiro 2013.

[Rai13d] Raindrip. Sistemas raindrip. http://raindrip.com/products?vmcchk=1, Janeiro 2013.

[Rai13e] Rainmaster. Sistemas rainmaster - icentral. http://www.rainmaster.com/PDF/iCentral_specification.pdf, Janeiro 2013.

[Sin08] Arun Kumar Singh. *Microcontroller and Embedded System*. New Age International, 1 edition, 2008.

[Spa12] SparkFun. Sm5100b. https://www.sparkfun.com/products/9607, Dezembro 2012.

[Spa13] SparkFun. Ftdi basic breakout - 3.3v. https://www.sparkfun.com/products/9873, Junho 2013.

[Tan06] Andrew S. Tanenbaum. *Structured Computer Organization*. Pearson Prentice Hall, 5 edition, 2006.

[Tor13] Toro. Sistemas toro - tricomm. http://www.toro.com/pt-pt/sports-fields-grounds/irrigation/central-control-systems/Pages/Model.aspx?pid=tricomm, Janeiro 2013.

[Tuc13] Tucor. Sistemas tucor. http://www.tucor.com/index-2.html, Janeiro 2013.

[Ubu] Ubuntu. Ubuntu server. http://www.ubuntu.com/server. Acedido em: 2013-08-02.

[VM] Duarte Vieira and Francisco Martins. Integrating wsn simulation into work ow testing and execution. http://www.di.fc.ul.pt/~fmartins/en/Publications_files/vieira.martins-integrating-WSN-WF.pdf.

[Wea13] Weathermatic. Sistemas weathermatic - smartlink. http://www.weathermatic.com/content/smartlink-network-property-owners, Janeiro 2013.

[Wik12] Wikipedia. At commands. http://en.wikipedia.org/wiki/AT_commands, Dezembro 2012.

Acrónimos

DIP Dual Inline Package.

EAR Enterprise ARchive.

EIS Enterprise Information System.

EJB Enterprise Java Bean.

GPRS General Packet Radio Service.

HTML HyperText Markup Language.

I2C Inter-Integrated Circuit.

ISP In-System Programming.

JavaEE Java Enterprise Edition.

JCA Java EE Connector Architecture.

JPA Java Persistence API.

JPQL Java Persistence Query Language.

JSP Java Server Pages.

MVC Model View Controller.

PCI Placa de Circuito Impresso.

POJO Plain Old Java Object.

PWM Pulse With Modulation.

RMI Remote Method Invocation.

RTC Real Time Clock.

SIM Subscriber Identity Module.

SMD Surface Mount Device.

SPI Serial Peripheral Interface.

SRI Sistema de Rega Inteligente.

TCP Transmission Control Protocol.

TDA Tipo de Dados Abstrato.

TQFP Plastic Quad Flat Package.

UART Universal Asynchronous Receiver and Transmitter.

URL Uniform Resource Locator.

USB Universal Serial Bus.

XML Extensible Markup Language.